Kreider, J.F.; Curtiss, P.S.; et. al. "Mechanical System Controls"
Mechanical Engineering Handbook
Ed. Frank Kreith
Boca Raton: CRC Press LLC, 1999

Mechanical System Controls

Jan F. Kreider
University of Colorado

Peter S. Curtiss
Architectural Energy Corporation

Thomas B. Sheridan
Massachusetts Institute of Technology

Shou-Heng Huang
Raytheon Co. Appliance Tech Center

Ron M. Nelson
Iowa State University

6.1 Human-Machine Interaction

Thomas B. Sheridan

Over the years machines of all kinds have been improved and made more reliable. However, machines typically operate as components of larger systems, such as transportation systems, communication systems, manufacturing systems, defense systems, health care systems, and so on. While many aspects of such systems can be and have been automated, the human operator is retained in many cases. This may be because of economics, tradition, cost, or (most likely) capabilities of the human to perceive patterns of information and weigh subtle factors in making control decisions which the machine cannot match.

Although the public as well as those responsible for system operation usually demand that there be a human operator, "human error" is a major reason for system failure. And aside from prevention of

error, getting the best performance out of the system means that human and machine must be working together effectively — be properly "impedance matched." Therefore, the performance capabilities of the human relative to those of the machine must be taken into account in system design.

Efforts to "optimize" the human-machine interaction are meaningless in the mathematical sense of optimization, since most important interactions between human and machine cannot be reduced to a mathematical form, and the objective function (defining what is good) is not easily obtained in any given context. For this reason, engineering the human-machine interaction, much as in management or medicine, remains an art more than a science, based on laboratory experiments and practical experience.

In the broadest sense, engineering the human-machine interface includes all of *ergonomics* or *human factors engineering,* and goes well beyond design of displays and control devices. Ergonomics includes not only questions of sensory physiology, whether or not the operator can see the displays or hear the auditory warnings, but also questions of *biomechanics*, how the body moves, and whether or not the operator can reach and apply proper force to the controls. It further includes the fields of operator selection and training, human performance under stress, human factors in maintenance, and many other aspects of the relation of the human to technology. This section focuses primarily on human-machine interaction in control of systems.

The human-machine interactions in control are considered in terms of Figure 6.1.1. In Figure 6.1.1a the human directly controls the machine; i.e., the control loop to the machine is closed through the physical sensors, displays, human senses (visual, auditory, tactile), brain, human muscles, control devices, and machine actuators. Figure 6.1.1b illustrates what has come to be called a *supervisory control system,* wherein the human intermittently instructs a computer as to goals, constraints, and procedures, then turns a task over to the computer to perform automatic control for some period of time.

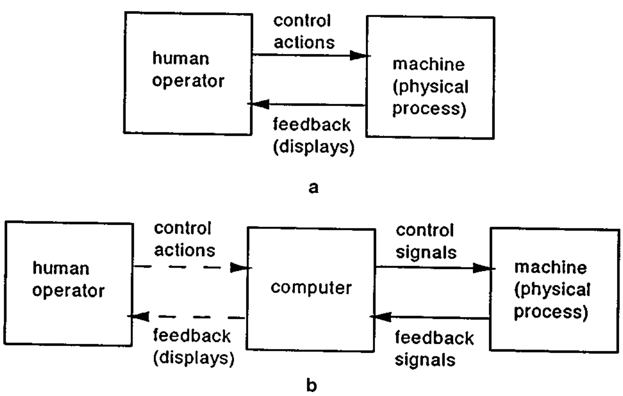


FIGURE 6.1.1 Direct manual control (a) and supervisory control (b).

Displays and control devices can be *analogic* (movement signal directions and extent of control action, isomorphic with the world, such as an automobile steering wheel or computer mouse controls, or a moving needle or pictorial display element). Or they can be *symbolic* (dedicated buttons or general-purpose keyboard controls, icons, or alarm light displays). In normal human discourse we use both speech (symbolic) and gestures (analogic) and on paper we write alphanumeric text (symbolic) and draw pictures (analogic). The system designer must decide which type of displays or controls best suits a particular application, and/or what mix to use. The designer must be aware of important criteria such as whether or not, for a proposed design, changes in the displays and controls caused by the human operator correspond in a natural and common-sense way to "more" or "less" of some variable as expected by that operator and correspond to cultural norms (such as reading from left to right in western countries), and whether or not the movement of the display elements correspond geometrically to movements of the controls.

Direct Manual Control

In the 1940s aircraft designers appreciated the need to characterize the transfer function of the human pilot in terms of a differential equation. Indeed, this is necessary for any vehicle or controlled physical process for which the human is the controller, see Figure 6.1.2. In this case both the human operator *H* and the physical process *P* lie in the closed loop (where *H* and *P* are Laplace transforms of the component transfer functions), and the *HP* combination determines whether the closed-loop is inherently stable (i.e., the closed loop characteristic equation $1 + HP = 0$ has only negative real roots).

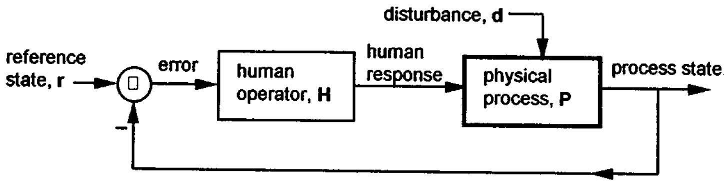


FIGURE 6.1.2 Direct manual control-loop analysis.

In addition to the stability criterion are the criteria of rapid response of process state *x* to a desired or reference state *r* with minimum overshoot, zero "steady-state error" between *r* and output *x*, and reduction to near zero of the effects of any disturbance input *d*. (The latter effects are determined by the closed-loop transfer functions $x = HP/(1 + HP)r + 1/(1 + HP)d$, where if the magnitude of *H* is large enough $HP/(1 + HP)$ approaches unity and $1/(1 + HP)$ approaches 0. Unhappily, there are ingredients of *H* which produce delays in combination with magnitude and thereby can cause instability. Therefore, *H* must be chosen carefully by the human for any given *P*.)

Research to characterize the pilot in these terms resulted in the discovery that the human adapts to a wide variety of physical processes so as to make $HP = K(1/s)(e^{-sT})$. In other words, the human adjusts *H* to make *HP* constant. The term *K* is an overall amplitude or gain, $(1/s)$ is the Laplace transform of an integrator, and (e^{-sT}) is a delay *T* long (the latter time delay being an unavoidable property of the nervous system). Parameters *K* and *T* vary modestly in a predictable way as a function of the physical process and the input to the control system. This model is now widely accepted and used, not only in engineering aircraft control systems, but also in designing automobiles, ships, nuclear and chemical plants, and a host of other dynamic systems.

Supervisory Control

Supervisory control may be defined by the analogy between a supervisor of subordinate staff in an organization of people and the human overseer of a modern computer-mediated semiautomatic control system. The supervisor gives human subordinates general instructions which they in turn may translate into action. The supervisor of a computer-controlled system does the same.

Defined strictly, *supervisory control* means that one or more human operators are setting initial conditions for, intermittently adjusting, and receiving high-level information from a computer that itself closes a control loop in a well-defined process through artificial sensors and effectors. For some time period the computer controls the process automatically.

By a less strict definition, *supervisory control* is used when a computer transforms human operator commands to generate detailed control actions, or makes significant transformations of measured data to produce integrated summary displays. In this latter case the computer need not have the capability to commit actions based upon new information from the environment, whereas in the first it necessarily must. The two situations may appear similar to the human supervisor, since the computer mediates both human outputs and human inputs, and the supervisor is thus removed from detailed events at the low level.

A supervisory control system is represented in Figure 6.1.3. Here the human operator issues commands to a *human-interactive* computer capable of understanding high-level language and providing integrated summary displays of process state information back to the operator. This computer, typically located in a control room or cockpit or office near to the supervisor, in turn communicates with at least one, and probably many (hence the dotted lines), *task-interactive* computers, located with the equipment they are controlling. The task-interactive computers thus receive subgoal and conditional branching information from the human-interactive computer. Using such information as reference inputs, the task-interactive computers serve to close low-level control loops between artificial sensors and mechanical actuators; i.e., they accomplish the low-level automatic control.

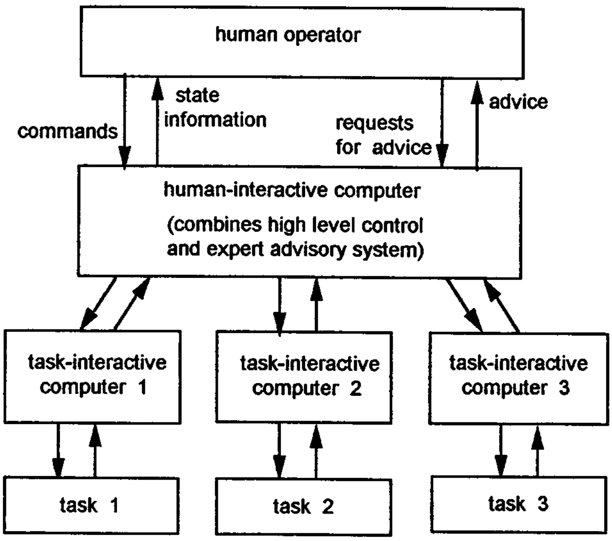


FIGURE 6.1.3 Supervisory control.

The low-level task typically operates at some physical distance from the human operator and his human-friendly display-control computer. Therefore, the communication channels between computers may be constrained by multiplexing, time delay, or limited bandwidth. The task-interactive computer, of course, sends analog control signals to and receives analog feedback signals from the controlled process, and the latter does the same with the environment as it operates (vehicles moving relative to air, sea, or earth, robots manipulating objects, process plants modifying products, etc.).

Supervisory command and feedback channels for process state information are shown in Figure 6.1.3 to pass through the left side of the human-interactive computer. On the right side are represented decision-aiding functions, with requests of the computer for advice and displayed output of advice (from a database, expert system, or simulation) to the operator. There are many new developments in computer-based decision aids for planning, editing, monitoring, and failure detection being used as an auxiliary part of operating dynamic systems. Reflection upon the nervous system of higher animals reveals a similar kind of supervisory control wherein commands are sent from the brain to local ganglia, and peripheral motor control loops are then closed locally through receptors in the muscles, tendons, or skin. The brain, presumably, does higher-level planning based on its own stored data and “mental models,” an internalized expert system available to provide advice and permit trial responses before commitment to actual response.

Theorizing about supervisory control began as aircraft and spacecraft became partially automated. It became evident that the human operator was being replaced by the computer for direct control responsibility, and was moving to a new role of monitor and goal-constraint setter. An added incentive was the U.S. space program, which posed the problem of how a human operator on Earth could control a

manipulator arm or vehicle on the moon through a 3-sec communication round-trip time delay. The only solution which avoided instability was to make the operator a supervisory controller communicating intermittently with a computer on the moon, which in turn closed the control loop there. The rapid development of microcomputers has forced a transition from manual control to supervisory control in a variety of industrial and military applications (Sheridan, 1992).

Let us now consider some examples of human-machine interaction, particularly those which illustrate supervisory control in its various forms. First, we consider three forms of vehicle control, namely, control of modern aircraft, “intelligent” highway vehicles, and high-speed trains, all of which have both human operators in the vehicles as well as humans in centralized traffic-control centers. Second, we consider telerobots for space, undersea, and medical applications.

Advanced Control of Commercial Aircraft

Flight Management Systems

Aviation has appreciated the importance of human-machine interaction from its beginning, and today exemplifies the most sophisticated forms of such interaction. While there have been many good examples of display and control design over the years, the current development of the flight management systems (FMS) is the epitome. It also provides an excellent example of supervisory control, where the pilot flies the aircraft by communicating in high-level language through a computer intermediary. The FMS is a centralized computer which interacts with a great variety of sensors, communication from the ground, as well as many displays and controls within the aircraft. It embodies many functions and mediates most of the pilot information requirements shown in Figure 6.1.4. Gone are the days when each sensor had its own display, operating independently of all other sensor-display circuits. The FMS, for example, brings together all of the various autopilot modes, from long-standing low-level control modes, wherein the aircraft is commanded to go to and hold a commanded altitude, heading, and speed, to more-sophisticated modes where the aircraft is instructed to fly a given course, consisting of a sequence of way points (latitudes and longitudes) at various altitudes, and even land automatically at a given airport on a given runway.

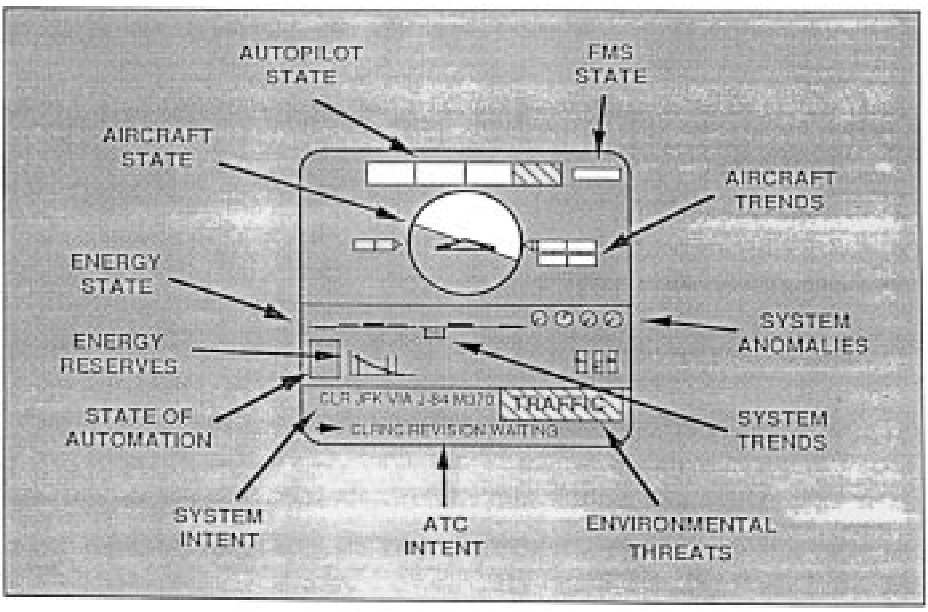


FIGURE 6.1.4 Pilot information requirements. (From Billings, 1991.)

Figure 6.1.5 illustrates one type of display mediated by the FMS, in this case integrating many formerly separate components of information. Mostly it is a multicolor plan-view map showing position and orientation of important objects relative to one's own aircraft (the triangle at the bottom). It shows heading (compass arc at top, present heading 175°), ground speed plus wind speed and wind direction (upper left), actual altitude relative to desired altitude (vertical scale on right side), programmed course connecting various way points (OPH and FLT), salient VOR radar beacons to the right and left of present position/direction with their codes and frequencies (lower left and right corners), the location of key VORs along the course (three-cornered symbols), the location of weather to be avoided (two gray blobs), and a predicted trajectory based on present turn rate, showing that the right turn is appropriately getting back on course.

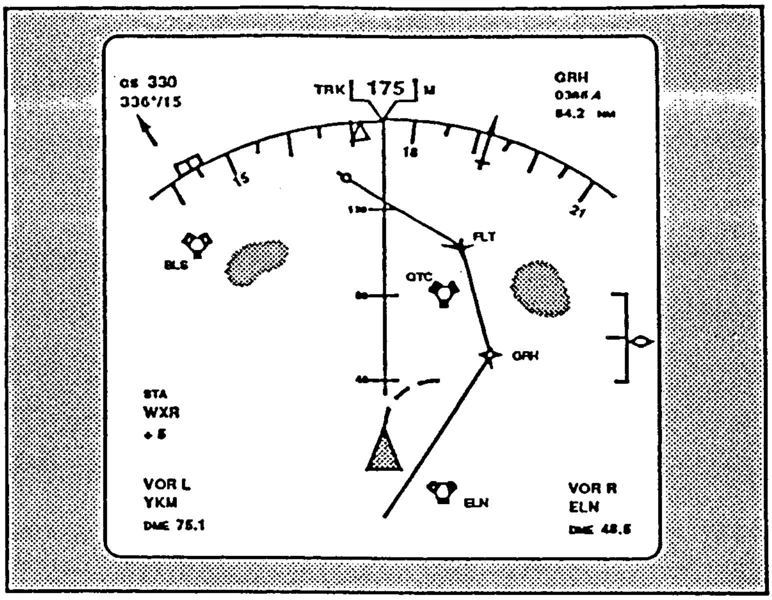


FIGURE 6.1.5 Integrated aircraft map display. (From Billings, 1991.)

Programming the FMS is done through a specialized keyboard and text display unit (Figure 6.1.6) having all the alphanumeric keys plus a number of special function keys. The displays in this case are specialized to the different phases of a flight (taxi, takeoff, departure, enroute approach, land, etc.), each phase having up to three levels of pages.

The FMS makes clear that designing displays and controls is no longer a matter of what can be built — the computer allows essentially any conceivable display/control to be realized. The computer can also provide a great deal of real-time advice, especially in emergencies, based on its many sensors and stored knowledge about how the aircraft operates. But pilots are not sure they need all the information which aircraft designers would like to give them, and have an expression "killing us with kindness" to refer to this plethora of available information. The question is what should be designed based on the needs and capabilities of the pilot.

Boeing, McDonnell Douglas, and Airbus have different philosophies for designing the FMS. Airbus has been the most aggressive in automating, intending to make piloting easier and safer for pilots from countries with less well established pilot training. Unfortunately, it is these most-automated aircraft which have had the most accidents of the modern commercial jets — a fact which has precipitated vigorous debate about how far to automate.

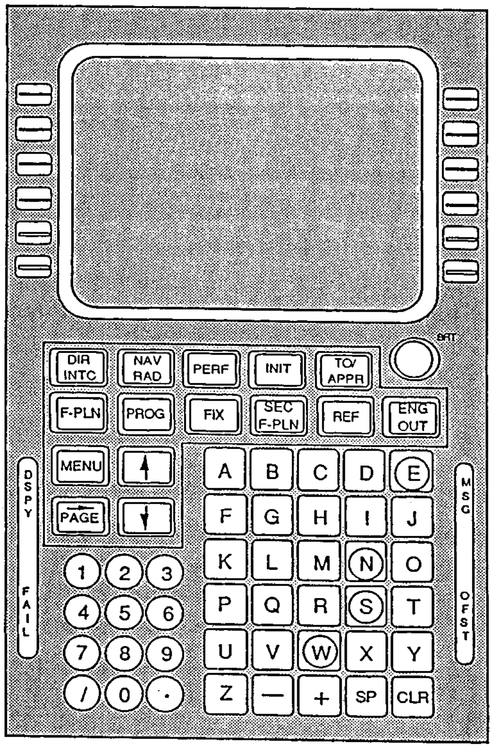


FIGURE 6.1.6 Flight management system control and display unit. (From Billings, 1991.)

Air Traffic Control

As demands for air travel continue to increase, so do demands for air traffic control. Given what are currently regarded as safe separation criteria, air space over major urban areas is already saturated, so that simply adding more airports is not acceptable (in addition to which residents do not want more airports, with their noise and surface traffic). The need is to reduce separations in the air, and to land aircraft closer together or on parallel runways simultaneously. This puts much greater demands on air traffic controllers, particularly at the terminal area radar control centers (TRACONs), where trained operators stare at blips on radar screens and verbally guide pilots entering the terminal airspace from various directions and altitudes into orderly descent and landing patterns with proper separation between aircraft.

Currently, many changes are being introduced into air traffic control which have profound implications for human-machine interaction. Where previously communication between pilots and air traffic controllers was entirely by voice, now digital communication between aircraft and ground (a system called *datalink)* allows both more and more reliable two-way communication, so that weather and runway and wind information, clearances, etc. can be displayed to pilots visually. But pilots are not so sure they want this additional technology. They fear the demise of the "party line" of voice communications with which they are so familiar and which permits all pilots in an area to listen in on each other's conversations.

New aircraft-borne radars allow pilots to detect air traffic in their own vicinity. Improved ground-based radars detect microbursts or wind shear which can easily put an aircraft out of control. Both types of radars pose challenges as to how best to warn the pilot and provide guidance as to how to respond. But they also pose a cultural change in air traffic control, since heretofore pilots have been dependent upon air traffic controllers to advise them of weather conditions and other air traffic. Furthermore, because of the new weather and collision-avoidance technology, there are current plans for radically altering the rules whereby high-altitude commercial aircraft must stick to well-defined traffic lanes. Instead, pilots

will have great flexibility as to altitude (to find the most favorable winds and therefore save fuel) and be able to take great-circle routes straight to their destinations (also saving fuel). However, air traffic controllers are not sure they want to give up the power they have had, becoming passive observers and monitors, to function only in emergencies.

Intelligent Highway Vehicles

Vehicle Guidance and Navigation Systems

The combination of GPS (global positioning system) satellites, high-density computer storage of map data, electronic compass, synthetic speech synthesis, and computer-graphic displays allows cars and trucks to know where they are located on the Earth to within 100 m or less, and can guide a driver to a programmed destination by a combination of a map display and speech. Some human factor challenges are in deciding how to configure the map (how much detail to present, whether to make the map north-up with a moving dot representing one's own vehicle position or current-heading-up and rapidly changing with every turn). The computer graphics can also be used to show what turns to anticipate and which lane to get in. Synthetic speech can reinforce these turn anticipations, can caution the driver if he is perceived to be headed in the wrong direction or off course, and can even guide him or her how to get back on course. An interesting question is what the computer should say in each situation to get the driver's attention, to be understood quickly and unambiguously but without being an annoyance. Another question is whether or not such systems will distract the driver's attention from the primary tasks, thereby reducing safety. The major vehicle manufacturers have developed such systems, they have been evaluated for reliability and human use, and they are beginning to be marketed in the United States, Europe, and Japan.

Smart Cruise Control

Standard cruise control has a major deficiency in that it knows nothing about vehicles ahead, and one can easily collide with the rear end of another vehicle if not careful. In a smart cruise control system a microwave or optical radar detects the presence of a vehicle ahead and measures that distance. But there is a question of what to do with this information. Just warn the driver with some visual or auditory alarm (auditory is better because the driver does not have to be looking in the right place)? Can a warning be too late to elicit braking, or surprise the driver so that he brakes too suddenly and causes a rear-end accident to his own vehicle. Should the computer automatically apply the brakes by some function of distance to obstacle ahead, speed, and closing deceleration. If the computer did all the braking would the driver become complacent and not pay attention, to the point where a serious accident would occur if the radar failed to detect an obstacle, say, a pedestrian or bicycle, or the computer failed to brake? Should braking be some combination of human and computer braking, and if so by what algorithm? These are human factor questions which are currently being researched.

It is interesting to note that current developmental systems only decelerate and downshift, mostly because if the vehicle manufacturers sell vehicles which claim to perform braking they would be open to a new and worrisome area of litigation.

The same radar technology that can warn the driver or help control the vehicle can also be applied to cars overtaking from one side or the other. Another set of questions then arises as to how and what to communicate to the driver and whether or not to trigger some automatic control maneuver in certain cases.

Advanced Traffic Management Systems

Automobile congestion in major cities has become unacceptable, and advanced traffic management systems are being built in many of these cities to measure traffic flow at intersections (by some combination of magnetic loop detectors, optical sensors, and other means), and regulate stoplights and message signs. These systems can also issue advisories of accidents ahead by means of variable message signs or radio, and give advice of alternative routings. In emergencies they can dispatch fire, police,

ambulances, or tow trucks, and in the case of tunnels can shut down entering traffic completely if necessary. These systems are operated by a combination of computers and humans from centralized control rooms. The operators look at banks of video monitors which let them see the traffic flow at different locations, and computer-graphic displays of maps, alarm windows, and textual messages. The operators get advice from computer-based expert systems, which suggest best responses based on measured inputs, and the operator must decide whether to accept the computer's advice, whether to seek further information, and how to respond.

High-Speed Train Control

With respect to new electronic technology for information sensing, storage, and processing, railroad technology has lagged behind that of aircraft and highway vehicles, but currently is catching up. The role of the human operator in future rail systems is being debated, since for some limited right-of-way trains (e.g., in airports) one can argue that fully automatic control systems now perform safely and efficiently. The train driver's principal job is speed control (though there are many other monitoring duties he must perform), and in a train this task is much more difficult than in an automobile because of the huge inertia of the train — it takes 2 to 3 km to stop a high-speed train. Speed limits are fixed at reduced levels for curves, bridges, grade crossings, and densely populated areas, while wayside signals temporarily command lower speeds if there is maintenance being performed on the track, if there are poor environmental conditions such as rock slides or deep snow, or especially if there is another train ahead. The driver must obey all speed limits and get to the next station on time. Learning to maneuver the train with its long time constants can take months, given that for the speed control task the driver's only input currently is an indication of current speed.

The author's laboratory has proposed a new computer-based display which helps the driver anticipate the future effects of current throttle and brake actions. This approach, based on a dynamic model of the train, gives an instantaneous prediction of future train position and speed based on current acceleration, so that speed can be plotted on the display assuming the operator holds to current brake-throttle settings. It also plots trajectories for maximum emergency braking and maximum service braking. In addition, the computer generates a speed trajectory which adheres at all (known) future speed limits, gets to the next station on time, and minimizes fuel/energy. Figure 6.1.7 shows the laboratory version of this display, which is currently being evaluated.

Telerobots for Space, Undersea, and Medicine

When nuclear power was first adopted in the late 1940s engineers began the development of master-slave remote manipulators, by which a human operator at one location could position and orient a device attached to his hand, and a servomechanism-controlled gripper would move in correspondence and handle objects at another location. At about the same time, remotely controlled wheeled vehicles, submarines, and aircraft began to be developed. Such manipulators and vehicles remotely controlled by humans are called *teleoperators*. Teleoperator technology got a big boost from the industrial robot technology, which came in a decade or so later, and provided improved vision, force, and touch sensors, actuators, and control software. Large teleoperators were developed for rugged mining and undersea tasks, and small teleoperators were developed for delicate tasks such as eye surgery. Eventually, teleoperators have come to be equipped with sensitive force feedback, so that the human operator not only can see the objects in the remote environment, but also can feel them in his grasp.

During the time of the Apollo flights to the moon, and stimulated by the desire to control lunar manipulators and vehicles from Earth and the fact that the unavoidable round-trip time delays of 3 sec (speed of light from Earth to moon and back) would not permit simple closed loop control, supervisory controlled teleoperators were developed. The human could communicate a subgoal to be reached and a procedure for getting there, and the teleoperator would be turned loose for some short period to perform automatically. Such a teleoperator is called a *telerobot.*

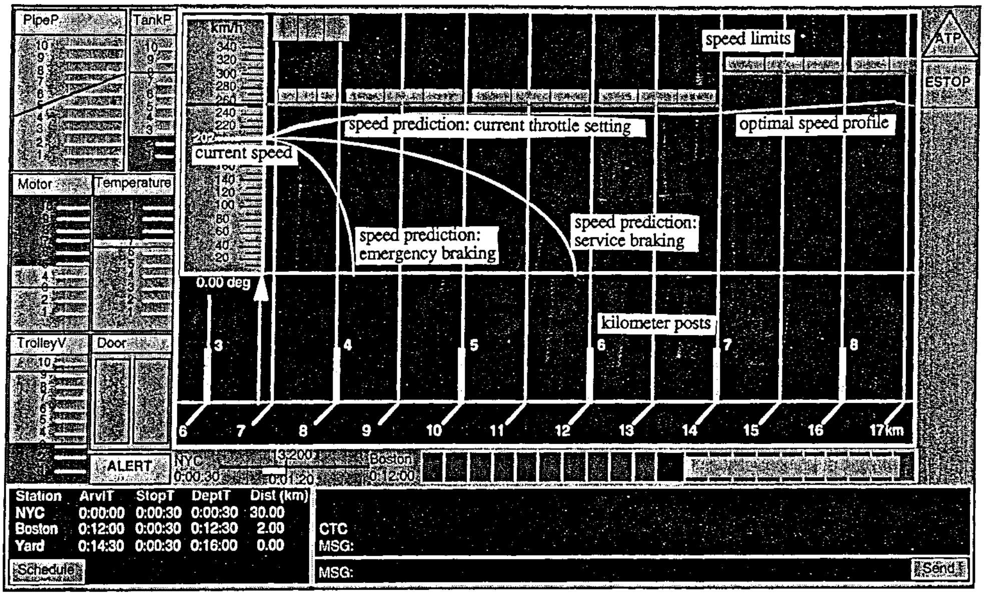


FIGURE 6.1.7 Prototype of computer-generated display for high speed trains. (From Askey, 1995.)

Figure 6.1.8 shows a the Flight Telerobotic Servicer (FTS) developed by Martin Marietta for the U.S. Space Station Freedom. It has two seven-degree of freedom (DOF) arms (including gripper) and one five-DOF "leg" for stabilizing itself while the arms work. It has two video "eyes" to present a stereoimage to its human operator. It can be configured either as a master-slave teleoperator (under direct human control) or as a telerobot (able to execute small programmed tasks using its own eyes and force sensors). Unfortunately, the FTS project was canceled by Congress.

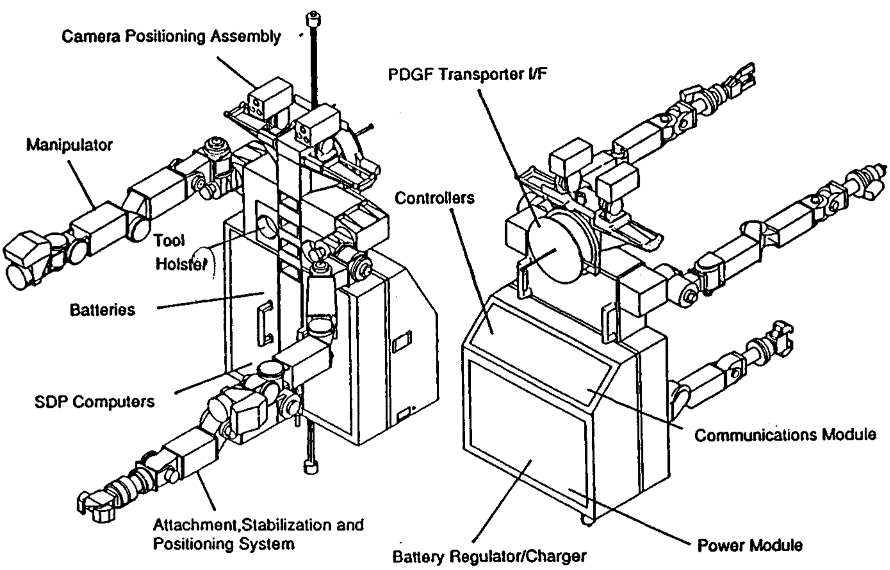


FIGURE 6.1.8 Flight Telerobotic Servicer prototype design. (Courtesy of NASA.)

Figure 6.1.9 shows the remotely operated submersible *Jason* developed by Woods Hole Oceanographic Institution. It is the big brother of *Jason Junior*, which swam into the interior of the ship *Titanic* and made a widely viewed video record when the latter was first discovered. It has a single manipulator arm, sonar and photo sensors, and four thrusters which can be oriented within limited range and which enable it to move in any direction. It is designed for depths up to 6000 m — rather severe pressures! It too, can be operated either in direct teleoperator mode or as a telerobot.

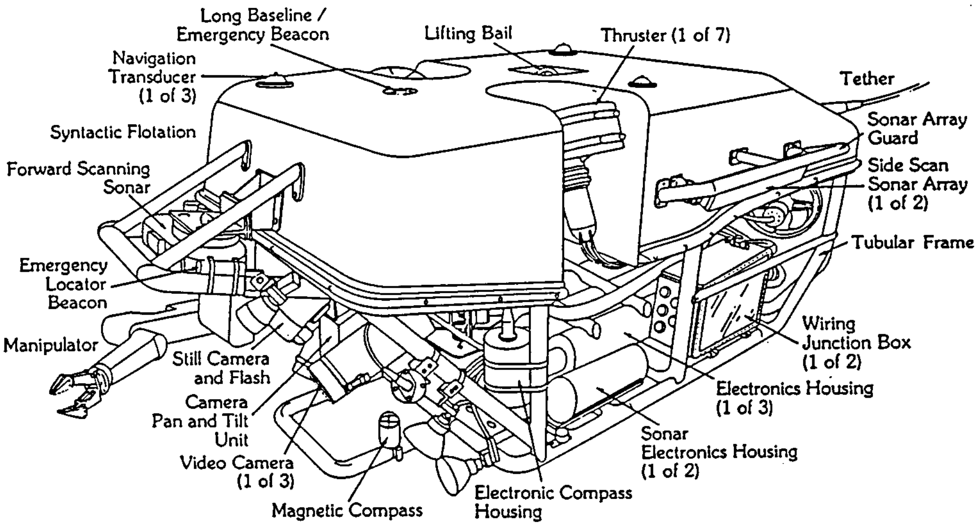


FIGURE 6.1.9 Deep ocean submersible *Jason*. (Courtesy of Woods Hole Oceanographic Institution.)

Common Criteria for Human Interface Design

Design of operator control stations for teleoperators poses the same types of problems as design of controls and displays for aircraft, highway vehicles, and trains. The displays must show the important variables unambiguously to whatever accuracy is required, but more than that must show the variables in relation to one another so as to clearly portray the current "situation" (situation awareness is currently a popular test of the human operator in complex systems). Alarms must get the operator's attention, indicate by text, symbol, or location on a graphic display what is abnormal, where in the system the failure occurred, what is the urgency, if response is urgent, and even suggest what action to take. (For example, the ground-proximity warning in an aircraft gives a loud "Whoop, whoop!" followed by a distinct spoken command "Pull up, pull up!") Controls — whether analogic joysticks, master-arms, or knobs — or symbolic special-purpose buttons or general-purpose keyboards — must be natural and easy to use, and require little memory of special procedures (computer icons and windows do well here). The placement of controls and instruments and their mode and direction of operation must correspond to the desired direction and magnitude of system response.

Human Workload and Human Error

As noted above, new technology allows combination, integration, and simplification of displays compared to the intolerable plethora of separate instruments in older aircraft cockpits and plant control rooms. The computer has taken over more and more functions from the human operator. Potentially these changes make the operator's task easier. However, it also allows for much more information to be presented, more extensive advice to be given, etc.

These advances have elevated the stature of the human operator from providing both physical energy and control, to providing only continuous control, to finally being a supervisor or a robotic vehicle or

system. Expert systems can now answer the operator's questions, much as does a human consultant, or whisper suggestions in his ear even if he doesn't request them. These changes seem to add many cognitive functions that were not present at an earlier time. They make the operator into a monitor of the automation, who is supposed to step in when required to set things straight. Unfortunately, people are not always reliable monitors and interveners.

Mental Workload

Under such complexity it is imperative to know whether or not the mental workload of the operator is too great for safety. Human-machine systems engineers have sought to develop measures of mental workload, the idea being that as mental load increases, the risk of error increases, but presumably measurable mental load comes before actual lapse into error.

Three approaches have been developed for measuring mental workload:

1. The first and most used is the subjective rating scale, typically a ten-level category scale with descriptors for each category from no load to unbearable load.
2. The second approach is use of physiological indexes which correlate with subjective scales, including heart rate and the variability of heart rate, certain changes in the frequency spectrum of the voice, electrical resistance of the skin, diameter of the pupil of the eye, and certain changes in the evoked brain wave response to sudden sound or light stimuli.
3. The third approach is to use what is called a secondary task, an easily measurable additional task which consumes all of the operator's attention remaining after the requirements of the primary task are satisfied. This latter technique has been used successfully in the laboratory, but has shortcomings in practice in that operators may refuse to cooperate.

Such techniques are now routinely applied to critical tasks such as aircraft landing, air traffic control, certain planned tasks for astronauts, and emergency procedures in nuclear power plants. The evidence suggests that supervisory control relieves mental load when things are going normally, but when automation fails the human operator is subjected rapidly to increased mental load.

Human Error

Human error has long been of interest, but only in recent decades has there been serious effort to understand human error in terms of categories, causation, and remedy. There are several ways to classify human errors. One is according to whether it is an error of *omission* (something not done which was supposed to have been done) or *commission* (something done which was not supposed to have been done). Another is *slip* (a correct intention for some reason not fulfilled) vs. a *mistake* (an incorrect intention which was fulfilled). Errors may also be classified according to whether they are in sensing, perceiving, remembering, deciding, or acting. There are some special categories of error worth noting which are associated with following procedures in operation of systems. One, for example, is called a *capture error,* wherein the operator, being very accustomed to a series of steps, say, A, B, C, and D, intends at another time to perform E, B, C, F. But he is "captured" by the familiar sequence B, C and does E, B, C, D.

As to effective therapies for human error, proper design to make operation easy and natural and unambiguous is surely the most important. If possible, the system design should allow for error correction before the consequences become serious. Active warnings and alarms are necessary when the system can detect incipient failures in time to take such corrective action. Training is probably next most important after design, but any amount of training cannot compensate for an error-prone design. Preventing exposure to error by guards, locks, or an additional "execute" step can help make sure that the most critical actions are not taken without sufficient forethought. Least effective are written warnings such as posted decals or warning statements in instruction manuals, although many tort lawyers would like us to believe the opposite.

Trust, Alienation, and How Far to Go with Automation

Trust

If operators do not trust their sensors and displays, expert advisory system, or automatic control system, they will not use it or will avoid using it if possible. On the other hand, if operators come to place too much trust in such systems they will let down their guard, become complacent, and, when it fails, not be prepared. The question of operator trust in the automation is an important current issue in human-machine interface design. It is desirable that operators trust their systems, but it is also desirable that they maintain alertness, situation awareness, and readiness to take over.

Alienation

There is a set of broader social effects that the new human-machine interaction can have, which can be discussed under the rubric of *alienation.*

1. People worry that computers can do some tasks much better than they themselves can, such as memory and calculation. Surely, people should not try to compete in this arena.
2. Supervisory control tends to make people remote from the ultimate operations they are supposed to be overseeing — remote in space, desynchronized in time, and interacting with a computer instead of the end product or service itself.
3. People lose the perceptual-motor skills which in many cases gave them their identity. They become "deskilled", and, if ever called upon to use their previous well-honed skills, they could not.
4. Increasingly, people who use computers in supervisory control or in other ways, whether intentionally or not, are denied access to the knowledge to understand what is going on inside the computer.
5. Partly as a result of factor 4, the computer becomes mysterious, and the untutored user comes to attribute to the computer more capability, wisdom, or blame than is appropriate.
6. Because computer-based systems are growing more complex, and people are being “elevated” to roles of supervising larger and larger aggregates of hardware and software, the stakes naturally become higher. Where a human error before might have gone unnoticed and been easily corrected, now such an error could precipitate a disaster.
7. The last factor in alienation is similar to the first, but all-encompassing, namely, the fear that a “race” of machines is becoming more powerful than the human race.

These seven factors, and the fears they engender, whether justified or not, must be reckoned with. Computers must be made to be not only “human friendly” but also not alienating with respect to these broader factors. Operators and users must become computer literate at whatever level of sophistication they can deal with.

How Far to Go with Automation

There is no question but that the trend toward supervisory control is changing the role of the human operator, posing fewer requirements on continuous sensory-motor skill and more on planning, monitoring, and supervising the computer. As computers take over more and more of the sensory-motor skill functions, new questions are being raised regarding how the interface should be designed to provide the best cooperation between human and machine. Among these questions are: To what degree should the system be automated? How much “help” from the computer is desirable? What are the points of diminishing returns?

Table 6.1.1 lists ten levels of automation, from 0 to 100% computer control. Obviously, there are few tasks which have achieved 100% computer control, but new technology pushes relentlessly in that direction. It is instructive to consider the various intermediate levels of Table 6.1.1 in terms not only of how capable and reliable is the technology but what is desirable in terms of safety and satisfaction of the human operators and the general public.

TABLE 6.1.1 Scale of Degrees of Automation

1.	The computer offers no assistance; the human must do it all.
2.	The computer offers a complete set of action alternatives, and
3.	Narrows the selection down to a few, or
4.	Suggests one alternative, and
5.	Executes that suggestion if the human approves, or
6.	Allows the human a restricted time to veto before automatic execution, or
7.	Executes automatically, then necessarily informs the human, or
8.	Informs the human only if asked, or
9.	Informs the human only if it, the computer, decides to
10.	The computer decides everything and acts autonomously, ignoring the human. The current controversy about how much to automate large commercial transport aircraft is often couched in these terms

Source: Sheridan 1987. With permission.

6.2 The Need for Control of Mechanical Systems

Peter S. Curtiss

Process control typically involves some mechanical system that needs to be operated in such a fashion that the output of the system remains within its design operating range. The objective of a process control loop is to maintain the process at the set point under the following dynamic conditions:

- The set point is changed;
- The load on the process is changed;
- The transfer function of the process is changed or a disturbance is introduced.

The Classical Control System Representation

Feedback-Loop System. A *feedback* (or *closed-loop*) system contains a process, a sensor and a controller. Figure 6.2.1 below shows some of the components and terms used when discussing feedback loop systems.

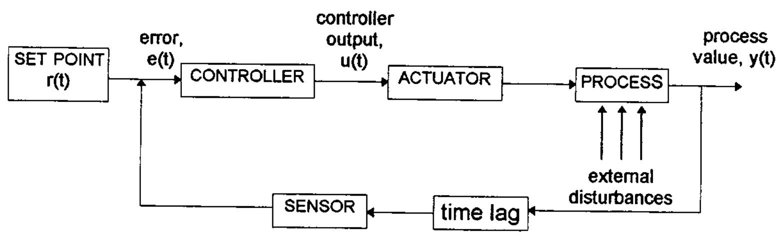


FIGURE 6.2.1 Typical feedback control schematic diagram.

Process. A *process* is a system that produces a motion, a temperature change, a flow, a pressure, or many other actions as a function of the actuator position and external inputs. The output of the process is called the process value. If a positive action in the actuator causes an increase in the process value then the process is called *direct acting*. If positive action in the actuator decreases the process value, it is called *reverse acting*.

Sensor. A *sensor* is a pneumatic, fluidic, or electronic or other device that produces some kind of signal indicative of the process value.

Set Point. The *set point* is the desired value for a process output. The difference between the set point and the process value is called the *process error.*

Controller. A *controller* sends signals to an actuator to effect changes in a process. The controller compares the set point and the process value to determine the process error. It then uses this error to adjust the output and bring the process back to the set point. The controller *gain* dictates the amount that the controller adjusts its output for a given error.

Actuator. An *actuator* is a pneumatic, fluidic, electric, or other device that performs any physical action that will control a process.

External Disturbances. An *external disturbance* is any effect that is unmeasured or unaccounted for by the controller.

Time Constants. The *time constant* of a sensor or process is a quantity that describes the dynamic response of the device or system. Often the time constant is related to the mass of an object or other dynamic effect in the process. For example, a temperature sensor may have a protective sheath around

it that must first be warmed before the sensor registers a change of temperature. Time constant can range from seconds to hours.

Dead Time. The *dead time* or *lag time* of a process is the time between the change of a process and the time this change arrives at the sensor. The delay time is not related to the time constant of the sensor, although the effects of the two are similar. Large dead times must be properly treated by the control system to prevent unstable control.

Hysteresis. *Hysteresis* is a characteristic response of positioning actuators that results in different positions depending on whether the control signal is increasing or decreasing.

Dead Band. The *dead band* of a process is that range of the process value in which no control action is taken. A dead band is usually used in two-position control to prevent "chattering" or in split-range systems to prevent sequential control loops from fighting each other.

Control Point. The *control point* is the actual, measured value of a process (i.e., the set point + steady-state offset + compensation).

Direct/Reverse Action. A *direct-acting* process will increase in value as the signal from the controller increases. A *reverse-acting* process will decrease in value as the signal from the controller increases.

Stability. The *stability* of a feedback control loop is an indication of how well the process is controlled or, alternatively, how controllable the process is. The stability is determined by any number of criteria, including overshoot, settling time, correction of deviations due to external disturbances, etc.

Electric Control. *Electric control* is a method of using low voltages (typically, 24 VAC) or line voltages (110 VAC) to measure values and effect changes in controlled variables.

Electronic Control. *Electronic controls* use solid-state, electronic components used for measurement and amplification of measured signals and the generation of proportional control signals.

Pneumatic Control. *Pneumatic controls* use compressed air as the medium for measuring and controlling processes.

Open-Loop Systems. An *open-loop system* is one in which there is no feedback. A whole-house attic fan in an example. It will continue to run even though the house may have already cooled off. Also, timed on/off devices are open loops.

Examples

Direct-Acting Feedback Control. A classic control example is a reservoir in which the fluid must be maintained at a constant level. Figure 6.2.2 shows this process schematically. The key features of this direct-acting system are labeled. We will refer to the control action of this system shortly after defining some terms.

Cascaded (Master-Slave) Control Loops. If a process consists of several subprocesses, each with a relatively different transfer function, it is often useful to use cascaded control loops. For example, consider a building housing a manufacturing line in which 100% outside air is used but which must also have very strict control of room air temperature. The room temperature is controlled by changing the position of a valve on a coil at the main air-handling unit that supplies the zone. Typically, the time constant of the coil will be much smaller than the time constant of the room. A single feedback loop would probably result in poor control since there is so much dead time involved with both processes. The solution is to use two controllers: the first (the master) compares the room temperature with the thermostat setting and sends a signal to the second (the slave) that uses that signal as its own set point for controlling the coil valve. The slave controller measures the output of the coil, not the temperature of the room. The controller gain on the master can be set lower than that of the slave to prevent excessive cycling.

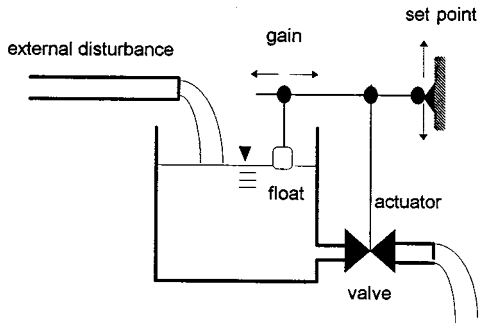


FIGURE 6.2.2 Example of a controlled process.

Sequential Control Loops. Sometimes control action is needed at more than one point in a process. An example of this is an air-handling unit that contains both heating and cooling coils in order to maintain a fixed outlet air temperature no matter the season. Typically, a *sequential* (or *split-range*) system in an air-handling unit will have three temperature ranges of operation, the first for heating mode, the last for cooling mode, and a middle dead-band region where neither the cooling nor heating coils are operating. Most sequential loops are simply two different control loops acting from the same sensor. The term *sequential* refers to the fact that in most of these systems the components are in series in the air or water stream.

Combined Feed-Forward/Feedback Loops. As pointed out earlier, feed-forward loops can be used when the effects of an external disturbance on a system are known. An example of this is outside air temperature reset control used to modify supply air temperatures. The control loop contains both a discharge air temperature sensor (the *primary* sensor) and an outdoor air temperature sensor (the *compensation* sensor). The designer should have some idea about the influence of the outside temperature on the heating load, and can then assign an *authority* to the effect of the outside air temperature on the controller set point. As the outdoor temperature increases, the control point decreases, and vice versa, as shown in Figure 6.2.3.

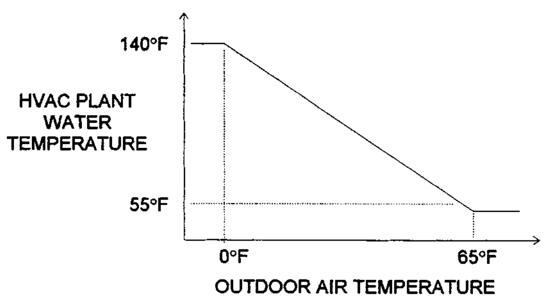


FIGURE 6.2.3 Example of the effect of compensation control.

Predictive Control. *Predictive control* uses a model of the process to predict what the process value will be at some point in the future based upon the current and past conditions. The controller then specifies a control action to be taken at the present that will reduce the future process error.

Adaptive Control. *Adaptive controllers* modify their gains dynamically so to adapt to current process conditions.

Supervisory Controllers. *Supervisory controllers* are used to govern the operation of an entire plant and/or control system. These may be referred to as *distributed control systems* (DCSs) which can be

used to govern the control of individual feedback loops and can also be used to ensure some kind of optimal performance of the entire plant. The controller will vary setpoints and operating modes in an attempt to minimize a cost function. A basic diagram of a supervisory controller in Figure 6.2.4.

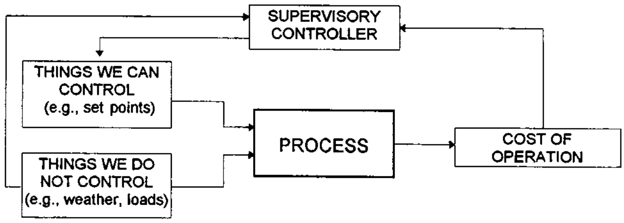


FIGURE 6.2.4 Typical supervisory controller.

6.3 Control System Analysis

Peter S. Curtiss

The Linear Process Approximation

To design controllers it is necessary to have both a dynamic process and control system representation. This section describes the key points of the most common such representation, that of linear processes and their controls. A process is basically a collection of mechanical equipment in which an input is changed or transformed somehow to produce an output. Many processes will be at near-steady-state, while others may be in a more or less constant state of change. We use building control systems as an illustration.

Steady-State Operation

The true response of a seemingly simple process can be, in fact, quite complex. It is very difficult to identify and quantify every single input because of the stochastic nature of life. However, practically any process can be approximated by an equation that takes into account the known input variables and produces a reasonable likeness to the actual process output.

It is convenient to use differential equations to describe the behavior of processes. For this reason, we will denote the "complexity" of the function by the number of terms in the corresponding differential equation (i.e., the *order* or *degree* of the differential equation). In a linear system analysis, we usually consider a step change in the control signal and observe the response. The following descriptions will assume a step input to the function, as shown in Figure 6.3.1. Note that a step change such as this is usually unlikely in most fields of control outside of electronic systems and even then can only be applied to a digital event, such as a power supply being switched on or a relay being energized. Zero-order system output has a one-to-one correspondence to the input,

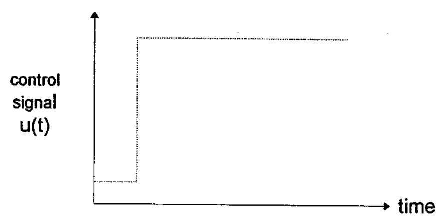


FIGURE 6.3.1 Step change in control signal.

$$y(t) = a_0 \cdot u(t)$$

First-order functions will produce a time-varying output with a step change as input,

$$\frac{dy(t)}{dt} + a_1 \cdot y(t) = b_1 \cdot u(t)$$

and higher-order functions will produce more complex outputs.

The function that relates the process value to the controller input is called the *transfer function* of the process. The time between the application of the step change, t_0, and the time at which the full extent of the change in the process value has been achieved is called the *transfer period*. A related phenomenon is process dead time. If there is a sufficient physical distance between the process output and the sensor assigned to measuring it, then one observes dead time during which the process output is not affected by the control signal (see Figure 6.3.2). The *process gain* (or *static gain*) is the ratio of the percentage

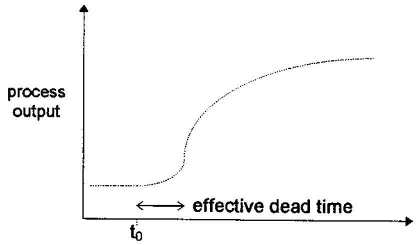


FIGURE 6.3.2 Effective dead time of a process subjected to a step change in controlled signal.

change of the process output to the corresponding percentage change of the control signal for a given response. For example, the gain can be positive (as in a heating coil) or negative (as in a cooling coil).

Dynamic Response

In practice, there are very few processes controlled in a stepwise fashion. Usually, the control signal is constantly modulating much the way that one makes small changes to the steering wheel of a car when driving down the highway. We now consider the dynamic process of level control in buckets filled with water (see Figure 6.3.3). Imagine that the level of water in the bucket on the left of Figure 6.3.3 is the control signal and the level of water in the bucket on the right is the process value. It is obvious that a step change in the control signal will bring about a first-order response of the process value.

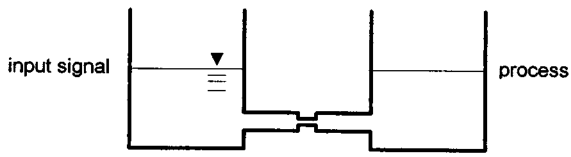


FIGURE 6.3.3 Connected water containers used for example of dynamic response.

Suppose, however, that a periodic signal is applied to the level of the bucket on the left. If the frequency of the signal is small enough, we see a response in the level in the bucket on the right that varies as a function of this driving force, but with a delay and a decrease in the amplitude.

Here the *dynamic process gain* is less than one even though the static process gain is one. There is no dead time in this process; as soon as we begin to increase the control signal the process value will also begin to increase. The dynamic process gain, therefore, can be defined similarly to that of the static gain — it is the ratio of the amplitude of the two signals, comparable with the normalized ranges used in the static gain definition.

The dynamic gain, as its name suggests, is truly dynamic. It will change not only according to the transfer function, but also to the frequency of the control signal. As the frequency increases, the output will lag even farther behind the input and the gain will continue to decrease. At one point, the frequency may be exactly right to cancel any past effects of the input signal (i.e., the phase shift is 180°) and the dynamic gain will approach zero. If the frequency rises further, the process output may decrease as the control signal increases (this can easily be the case with a building cooling or heating coil due to the mass effects) and the dynamic gain will be negative!

At this point it is convenient to define a feedback loop mathematically. A general feedback loop is shown in Figure 6.3.4. The controller, actuator, and process have all been combined into the *forward transfer function* (or *open-loop transfer function*) G and the sensor and dead time have all been combined into the *feedback path* transfer function H. The overall *closed-loop transfer function* is defined as

$$\frac{C}{R} = \frac{C}{1 + G \cdot H}$$

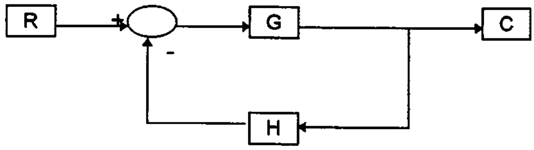


FIGURE 6.3.4 Generalized feedback loop.

The right-hand side of this equation is usually a ratio of two polynomials when using Laplace or *z*-transforms. The roots of the numerator are called the *zeros* of the transfer function and the roots of the denominator are called the *poles* (Shinners, 1978).

The denominator of the closed loop transfer function, $1 + G \cdot H$, is called the *characteristic function.* If we set the characteristic function equal to zero we have the *characteristic equation*

$$1 + G \cdot H = 0$$

The characteristic equation can be used to assess process control stability during system design.

Representation of Processes in *t*, *s*, and *z* Domains

We cannot hope to ever know how a process truly behaves. The world is an inherently stochastic place and any model of a system is going to approximate at best. Nonetheless, we will need to choose some kind of representation in order to perform any useful analysis.

This section will consider three different domains: the continuous-time domain, the frequency domain, and the discrete-time domain. The frequency domain is useful for certain aspects of controller design, whereas the discrete-time domain is used in digital controllers.

Continuous-Time-Domain Representation of a Process

In the time domain we represent a process by a differential equation, such as

$$\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + a_2 \frac{d^{n-2} y}{dt^{n-2}} + \cdots + a_{n-1} \frac{dy}{dt} + a_n y = b_0 \frac{d^m u}{dt^m} + b_1 \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_{m-1} \frac{du}{dt} + b_m u$$

This is just a generalization of the first-order system equation described earlier.

Frequency-Domain Representation of a Process — Laplace Transforms

The solution of higher-order system models, closed-form solution is difficult in the time domain. For this reason, process transfer functions are often written using Laplace transforms. A Laplace transform is a mapping of a continuous-time function to the frequency domain and is defined as

$$F(s) = \int_0^\infty f(t) e^{-st}\, dt$$

Laplace transforms are treated in Section 19. This formulation allows us to greatly simplify problems involving ordinary differential equations that describe the behavior of systems. A transformed differential equation becomes purely algebraic and can be easily manipulated and solved. These solutions are not of great interest in themselves in modern control system design but the transformed system (+) controller differential equation is very useful in assessing control stability. This is the single key aspect of Laplace transforms that is of most interest. Of course, it is possible just to solve the governing differential equation for the system directly and explore stability in that fashion.

The Laplace transform of the previous differential equation is

$$s^n Y(s) + A_1 s^{n-1} Y(s) + \dots + A_{n-1} sY(s) + A_n Y(s) = B_0 s^m U(s) + B_1 s^{n-1} U(s) + \dots + B_{n-1} sU(s) + B_n U(s)$$

This equation can be rewritten as

$$Y(s) \cdot \left(s^n + A_1 s^{n-1} + \dots + A_{n-1} s + A_n\right) = U(s) \cdot \left(B_0 s^m + B_1 s^{n-1} + \dots + B_{n-1} s + B_n\right)$$

so that the transfer function is found from

$$\frac{Y(s)}{U(s)} = \frac{s^m + B_1 s^{m-1} + \dots + B_{m-1} s + A_m}{s^n + A_1 s^{n-1} + \dots + A_{n-1} s + A_n}$$

This is the expression that is used for stability studies.

Discrete-Time-Domain Representation of a Process. A process in the discrete time domain is described (Radke and Isermann, 1989) by

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + a_3 y(k-3) + \dots + b_1 u(k-1) + b_2 u(k-2) + b_3 u(k-3) + \dots$$

This representation is of use when one is designing and analyzing the performance of direct digital control (DDC) systems. Note that the vectors **a** and **b** are *not* the same as for the continuous-time domain equation. The *z*-transform uses the backward shift operator and therefore the *z*-transform of the discrete-time equation is given by

$$y\left(1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} + \dots\right) = u\left(b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3} + \dots\right)$$

The transfer function can now be found:

$$\frac{y}{u} = \frac{b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3} + \dots}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} + \dots}$$

z-Transform Details. Because *z*-transforms are important in modern control design and are not treated elsewhere in this handbook, some basics of their use are given below. More and more control applications are being turned over to computers and DDC systems. In such systems, the sampling is not continuous, as required for a Laplace transform. The control loop schematic is shown in Figure 6.3.5.

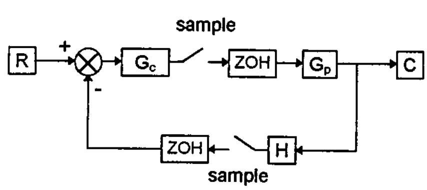


FIGURE 6.3.5 Sampled feedback loop.

It would be prohibitively expensive to include a voltmeter or ohmmeter on each loop; therefore, the controller employs what is called a *zero-order hold*. This basically means that the value read by the controller is "latched" until the next value is read in. This discrete view of the world precludes the use of Laplace transforms for analyses and makes it necessary, therefore, to find some other means of

simplifying the simulation of processes and controllers. The following indicates briefly how *z*-transforms of controlled processes can be derived and how they are used in a controls application. In the design section of this chapter we will use the *z*-transform to assess controller stability.

Recall that the Laplace transform is given as

$$\mathscr{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}\,dt$$

Now suppose we have a process that is sampled at a discrete, constant time interval *T*. The index *k* will be used to count the intervals,

$$\text{at time } t = 0,\ k = 0,$$

$$\text{at time } t = T,\ k = 1,$$

$$\text{at time } t = 2T,\ k = 2,$$

$$\text{at time } t = 3T,\ k = 3,$$

and so forth. The equivalent Laplace transform of a process that is sampled at a constant interval *T* can be represented as

$$\mathscr{L}\{f^*(t)\} = \sum_{k=0}^{\infty} f(kT)e^{-s\cdot kT}$$

By substituting the *backward-shift operator* z for e^{Ts}, we get the definition of the *z*-transform:

$$Z\{f(t)\} = \sum_{k=0}^{\infty} f(kT)z^{-k}$$

Example of Using z-Transfer Functions. Suppose we have a cylindrical copper temperature sensor in a fluid stream with material properties as given. We wish to establish its dynamic characteristics for the purpose of including it in a controlled process model using both Laplace and *z*-transforms. Figure 6.3.6 shows the key characteristics of the sensor. The sensor measures 0.5 cm in diameter and is 2 cm long. For the purposes of this example we will assume that the probe is solid copper. The surface area of the sensor is then

$$A_s = 2\cdot\pi\cdot(0.25\text{ cm})^2 + \pi\cdot(0.5\text{ cm})\cdot(2\text{ cm}) \approx 3.5\text{ cm}^2$$

and the mass is

$$M_s = \left(9\text{ g/cm}^3\right)\cdot\left[(2\text{ cm})\cdot\pi\cdot(0.25\text{ cm})^2\right] \approx 3.5\text{ g}$$

The thermal capacitance of the sensor is found from the product of the mass and the heat capacity,

$$C_s = M_s\cdot c_p = 3.5\text{ g}\cdot 0.4\text{ J/g}\cdot\text{K} = 1.4\text{ J/K}$$

and the total surface heat transfer rate is the product of the area and the surface heat transfer coefficient,

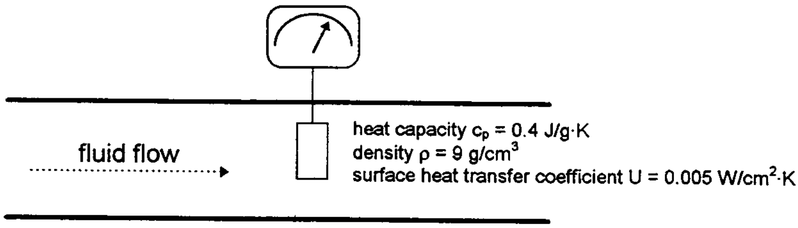


FIGURE 6.3.6 Fluid temperature sensor used in example.

$$UA_s = 0.005\ \text{W/cm}^2 \cdot \text{K} \cdot 3.5\ \text{cm}^2 = 0.018\ \text{W/K}$$

Now we can perform an energy balance on the sensor by setting the sum of the energy flow into the sensor and the energy stored in the sensor equal to zero:

$$C_s = \frac{dT_s}{dt} + \left(T_s - T_a\right) \cdot UA_s = 0$$

where T_s is the temperature of the sensor and T_a is the ambient fluid temperature. This relationship is a nonhomogeneous first-order differential equation:

$$\frac{dT_s}{dt} + \frac{UA_s}{C_s} T_s = \frac{UA_s}{C_s} T_a$$

The *time constant* of the sensor is defined as

$$\tau = \frac{C_s}{UA_s} = \frac{1.4\ \text{J/K}}{0.018\ \text{W/K}} \approx 80\ \text{sec}$$

The differential equation that describes this sensor is

$$\frac{dT_s}{dt} + \frac{1}{\tau} T_s = \frac{1}{\tau} T_a$$

This example will find the response of the sensor when the fluid temperature rises linearly by 30°C from time $t = 0$ to time $t = 200$ seconds and then remains constant. That is, the driving function is

$$T_a(t) = \frac{30°\text{C}}{200\ \text{sec}} t = 0.15°\text{C/sec}\ t \qquad 0 \le t < 200\ \text{sec}$$

$$T_a(t) = 30°\text{C} \qquad t \ge 200\ \text{sec}$$

Assuming an initial condition of $T_s = 0$ at $t = 0$, we find that

$$T_s(t) = 0.15t - 12 + 12e^{-0.0125t} \qquad \text{for} \quad t < 200\ \text{sec}$$

and

$$T_s(t) = 30 - 134.2e^{-t/\tau} \qquad \text{for} \quad t > 200\ \text{sec}$$

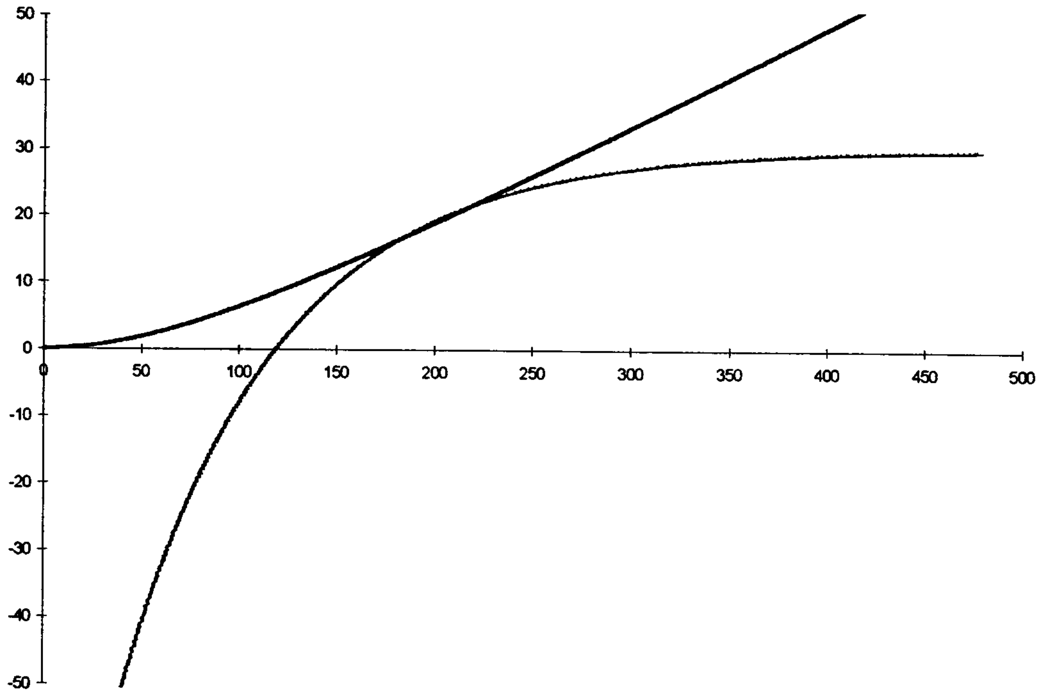


FIGURE 6.3.7 Time domain solution of example.
First line: $T = 0.15t - 12 + 12e^{-0.0125t}$. Second line: $T = 30 - 134.2e^{-t/r}$.

A graph of the entire process (rise time and steady state) is shown in Figure 6.3.7. The two lines intersect at $t = 200$ seconds.

<u>Solution of example in frequency domain</u>

This same process can be solved using Laplace transforms. First we will solve this problem for the first 200 sec. The Laplace transform of a ramp function can be found from the tables (see Section 19) to get

$$T_s(s) = \frac{1}{\tau(s + 1/\tau)} \frac{0.15}{s^2}$$

This can be expanded by partial fractions to get

$$T_s(s) = 0.15\left[\frac{\tau}{s} + \frac{1}{s^2} + \frac{\tau}{s + 1/\tau}\right]$$

Using the table for the inverse Laplace transforms gives

$$T_s(t) = 0.15\left[-\tau + t + \tau e^{-t/\tau}\right]$$

Substituting $\tau = 80$,

$$T_s(t) = -12 + 0.15t + 12e^{-0.0125t}$$

This is exactly the same as the time-domain solution above. For the steady-state driving force of T_a = 30 for t > 200 sec we also find the same result.

Solution of example in the discrete-time domain

We consider the same problem in the discrete-time domain that a DDC system might use. Recall that the transfer function in the frequency domain was given by

$$\frac{T_s(s)}{T_a(s)} = \frac{\frac{1}{\tau}}{s + \frac{1}{\tau}}$$

We look to the table in this book's appendix and find that the discrete-time equivalent is

$$\frac{T_s(z)}{T_a(z)} = \frac{1}{\tau} \cdot \frac{z}{z - e^{-T/\tau}}$$

where T is the sampling frequency in seconds. The driving function T_a is a given as

$$T_a(t) = \alpha t$$

where α is the rate of change of the temperature as above (0.15°C/sec). Note that to put this into the discrete-time domain we must correct this rate by the sampling interval,

$$T_a(z) = 0.15 \cdot T \cdot \frac{Tz}{(z-1)^2}$$

using α = (0.15 · T) ºC/sampling interval.

We can now express the response of the process as

$$T_s(z) = \frac{1}{\tau} \cdot \left(\frac{z}{z - e^{-T/\tau}} \right) \cdot (0.15 \cdot T) \cdot \left(\frac{Tz}{(z-1)^2} \right)$$

since the z operator acts on the sensed temperature by performing a backward shift of the time index. In other words, the previous equation can be rewritten as

$$T_{s,k} - \left(2 + e^{-T/\tau}\right) T_{s,k-1} + \left(1 + 2e^{-T/\tau}\right) T_{s,k-2} - e^{-T/\tau} T_{s,k-3} = \frac{0.15}{\tau} \cdot T^2 z^{-1}$$

So the current temperature is determined by the previous three temperature measurements,

$$T_{s,k} = \left(2 + e^{-T/\tau}\right) T_{s,k-1} - \left(1 + 2e^{-T/\tau}\right) T_{s,k-2} + e^{-T/\tau} T_{s,k-3} + \frac{0.15}{\tau} \cdot T^2 z^{-1} \qquad \text{for } kT < 200 \text{ sec}$$

Regarding the last term in the equation, recall that the inverse transform of z^{-k} is given as 1 when $t = k$, and zero otherwise. This term provides the initial "jump-start" of the progression. Table 6.3.1 shows the first few time steps for the z-domain solution using time steps of 0.1 and 1.0 seconds. In general, the accuracy of the z-domain solution increases as the time step grows smaller. The solution for kT > 200 seconds is similar to that for the Laplace transforms.

TABLE 6.3.1 Initial Time Steps for *z*-Domain Solution of Example

Initial 10 Steps for T = 0.1 sec				Initial 10 Steps for T = 1.0 sec			
k	Time	$T_{s,exact}$	$T_{s,z\ transform}$	k	Time	$T_{s,exact}$	$T_{s,z\ transform}$
0	0.00	0.00000	0.00000	0	0.00	0.0000	0.0000
1	0.10	0.00001	0.00002	1	1.00	0.0009	0.0019
2	0.20	0.00004	0.00006	2	2.00	0.0037	0.0056
3	0.30	0.00008	0.00011	3	3.00	0.0083	0.0112
4	0.40	0.00015	0.00019	4	4.00	0.0148	0.0185
5	0.50	0.00023	0.00028	5	5.00	0.0230	0.0277
6	0.60	0.00034	0.00039	6	6.00	0.0329	0.0386
7	0.70	0.00046	0.00052	7	7.00	0.0446	0.0512
8	0.80	0.00060	0.00067	8	8.00	0.0580	0.0656
9	0.90	0.00076	0.00084	9	9.00	0.0732	0.0816
10	1.00	0.00093	0.00103	10	10.00	0.0900	0.0994

The next three figures show the effect of using different time intervals in the *z*-domain solution. The values shown here are for the example outlined in this section. If one could use an infinitesimally small time step, the *z*-domain solution would match the exact solution. Of course, this would imply a much larger computational effort to simulate even a small portion of the process. In practice, a time interval will be chosen that reflects a compromise between accuracy and speed of calculation.

Each graph shows two lines, one for the exact solution of the first 200 sec of the example and the other for the *z*-domain solution. Figures 6.3.8 to 6.3.10 give the *z*-domain solution using time intervals of 0.1, 1.0 and 10.0 seconds, respectively. Notice that there is not much difference between the first two graphs even though there is an order of magnitude difference between the time intervals used. The latter two graphs show significant differences.

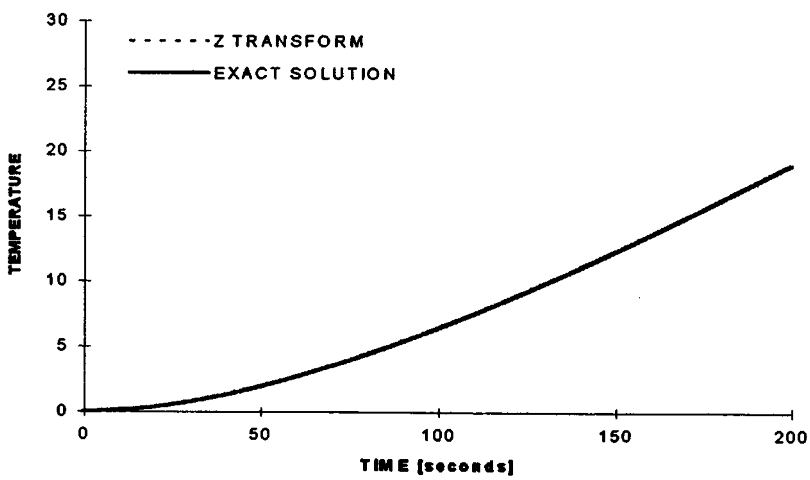


FIGURE 6.3.8 Result of *z*-transform when T = 0.1 sec.

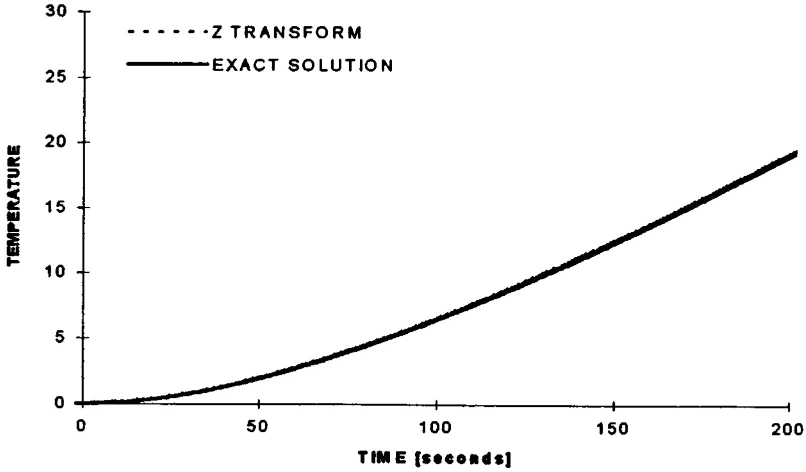


FIGURE 6.3.9 Results of *z*-transform when T = 1.0 sec.

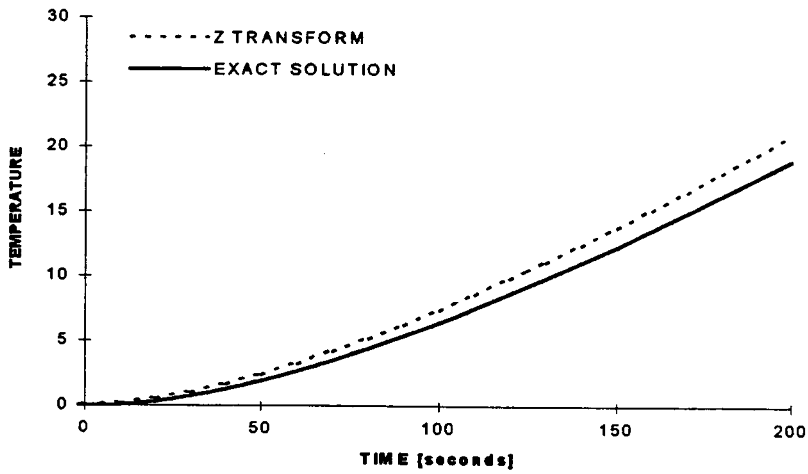


FIGURE 6.3.10 Results of *z*-transform when T = 10.0 sec.

6.4 Control System Design and Application

Peter S. Curtiss

Controllers

Controllers are akin to processes in that they have gains and transfer functions. Generally, there is no dead time in a controller or it is so small as to be negligible.

Steady-State Effects of Controller Gain

Recall that the process static gain can be viewed as the total change in the process value due to a 100% change in the controller output. A proportional controller acts like a multiplier between an *error signal* and this process gain. Under stable conditions, therefore, there must be some kind of error to yield any controller output. This is called the steady-state or static *offset.*

Dynamic Effects of Controller Gain

Ideally, a controller gain value is chosen that compensates for the dynamic gain of the process under normal operating conditions. The total loop dynamic gain can be considered as the product of the process, feedback, and controller gains. If the total dynamic loop gain is one, the process will oscillate continuously at the natural frequency of the loop with no change in amplitude of the process value. If the loop gain is greater than one, the amplitude will increase with each cycle until the limits of the controller or process are reached or until something fails. If the dynamic loop gain is less than one, the process will eventually settle down to stable control.

Controller Bias

The controller bias is a constant offset applied to the controller output. It is the output of the controller if the error is zero,

$$u = K \cdot e + M$$

where M is the bias. This is useful for processes that become nonlinear at the extremes or for process in which the normal operating conditions are at a nonzero controller output.

PID Controllers

Many mechanical systems are controlled by proportional-integral-derivative (PID) controllers. There are many permutations of such controllers which use only certain portions of the PID controllers or use variations of this kind of controller. In this section we consider this very common type of controller.

Proportional Control

Proportional control results in action that is linear with the error (recall the error definition in Figure 6.2.1) The proportional term, $K_p \cdot e$, has the greatest effect when the process value is far from the desired setpoint. However, very large values of K_p will tend to force the system into oscillatory response. The proportional gain effect of the controller goes to zero as the process approaches set point. Purely proportional control should therefore only be used when

- The time constant of the process is small and hence a large controller gain can be used;
- The process load changes are relatively small so that the steady-state offset is limited;
- The steady-state offset is within an acceptable range.

Integral Control

Integral control makes a process adjustment based on the cumulative error, not its current value. The integral term K_i is the reciprocal of the reset time, T_r, of the system. The reset time is the duration of

each error-summing cycle. Integral control can cancel any steady-state offsets that would occur when using purely proportional control. This is sometimes called *reset* control.

Derivative Control

Derivative control makes a process adjustment based on the current rate of change of the process control error. Derivative control is typically used in cases where there is a large time lag between the controlled device and the sensor used for the feedback. This term has the overall effect of preventing the actuator signal from going too far in one direction or another, and can be used to limit excessive overshoot.

PID Controller in Time Domain

The PID controller can be represented in a variety of ways. In the time domain, the output of the controller is given by

$$u(t) = K_p \left[e(t) + K_i \int_0^t e(t)\, dt + K_d \frac{de(t)}{dt} \right]$$

PID Controller in the *s* Domain

It is relatively straightforward to derive the Laplace transform of the time-domain PID equation. The transfer function of the controller is

$$\frac{U(s)}{E(s)} = \left[K_p + \frac{K_p K_i}{s} + K_p K_d s \right]$$

This controller transfer function can be multiplied by the process transfer function to yield the overall forward transfer function **G** of an *s*-domain process model. The criteria described earlier can then be used to assess overall system stability.

PID Controller in the *z* Domain

Process data are measured discretely at time intervals Δt, and the associated PID controller can be represented by

$$u(k) = K_p \left[e(k) + K_i\, \Delta t \sum_{i=0}^{k} e(i) + K_d \frac{e(k) - e(k-1)}{\Delta t} \right]$$

The change of the output from one time step to the next is given by $u(k) - u(k-1)$, so the PID *difference equation* is

$$u(k) - u(k-1) = K_p \left[\left(1 + \frac{K_d}{\Delta t}\right) e(k) + \left(K_i\, \Delta t - 1 - 2\frac{K_d}{\Delta t}\right) e(k-1) + \left(\frac{K_d}{\Delta t}\right) e(k-2) \right]$$

and can be simplified as

$$u(k) - u(k-1) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2)$$

where

$$q_0 = K_p\left(1 + \frac{K_d}{\Delta t}\right); \qquad q_1 = K_p\left(K_i\, \Delta t - 1 - 2\frac{K_d}{\Delta t}\right); \qquad q_2 = K_p\left(\frac{K_d}{\Delta t}\right)$$

Note that we can write this as

$$u\left(1-z^{-1}\right)=e\left(q_0+q_1z^{-1}+q_2z^{-2}\right)$$

The z-domain transfer function of the PID controller is then given as

$$\frac{u(z)}{e(z)}=\frac{q_0+q_1z^{-1}+q_2z^{-2}}{1-z^{-1}}=\frac{q_0z^2+q_1z+q_2}{z^2-z}$$

Controller Performance Criteria and Stability

Performance Indexes

Obviously, in feedback loops we wish to reduce the process error quickly and stably. The control systems engineer can use different cost functions in the design of a given controller depending on the criteria for the controlled process. Some of these cost functions (or *performance indexes*) are listed here:

ISE	Integral of the square of the error	$\int^2 e$
ITSE	Integral of the time and the square of the error	$\int e^2$
ISTAE	Integral of the square of the time and the absolute error	$\int^2 \lvert e\rvert$
ISTSE	Integral of the square of the time and the square of the error	$\int^2 e^2$

These indexes are readily calculated with DDC systems and can be used to compare the effects of different controller settings, gains, and even control methods.

Stability

Stability in a feedback loop means that the feedback loop will tend to converge on a value as opposed to exhibiting steady-state oscillations or divergence. Recall that the closed-loop transfer function is given by

$$\frac{C}{R}=\frac{G}{1+GH}$$

and that the denominator, $1 + GH$, when equated to zero, is called the characteristic equation. Typically, this equation will be a polynomial in s or z depending on the method of analysis of the feedback loop. Two necessary conditions for stability are that all powers of s must be present in the characteristic equation from zero to the highest order and that all coefficients in the characteristic equation must have the same sign. Note that the process may still be unstable even when these conditions are satisfied.

Roots of the Characteristic Equation. The roots of the characteristic equation play an important role in determining the stability of a process. These roots can be real and/or imaginary and can be plotted as shown in Figure 6.4.1. In the s-domain, if all the roots are in the left half-plane (i.e., to the left of the imaginary axis), then the feedback loop is guaranteed to be asymptotically stable and will converge to a single output value. If one or more roots are in the right half-plane, then the process is unstable. If one or more roots lie on the imaginary axis and none are in the right half-plane, then the process is considered to be marginally stable. In the z-domain, if all the roots lie within the unit circle about the origin then the feedback loop is asymptotically stable and will converge. If one or more roots lie outside

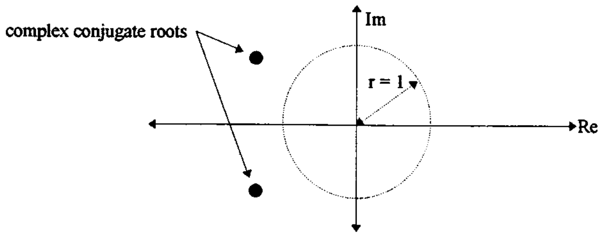


FIGURE 6.4.1 Placement of roots in the imaginary plane (showing unit circle).

the unit circle then the process is unstable. If one or more roots lie on the unit circle and none are outside the unit circle, then the process is marginally stable.

Root locus example

Consider the feedback loop shown in Figure 6.4.2. The characteristic equation is given by 1 + *GH* = 0 or

$$1+K\left(\frac{s}{s+\alpha}\right)\left(\frac{1}{s+\beta}\right)=0$$

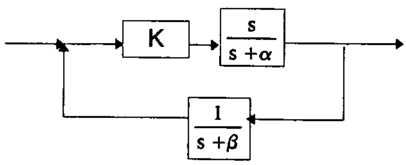


FIGURE 6.4.2 Simple feedback control loop.

For different values of *K* we can plot the roots of this equation. The graph in Figure 6.4.3 shows an example plot when the characteristic equation is given by $s^2 + (1.25 + K)s + 1.25 = 0$. The plot shows that a system described by this characteristic demonstrates stable response for a process gain of $0.0 \leq K \leq 10.0$. For gains greater than 10, there exists at least one root in the right half-plane and the process is not under stable control.

Note that the root locus plot is always symmetric about the real axis and that the number of separate segments of the locus is equal to the number of roots of the characteristic equation (i.e., the number of poles of the closed-loop transfer function).

Routh-Hurwitz Stability Criteria. The Routh-Hurwitz method is an tabular manipulation of the characteristic equation in the frequency domain and is used to assess stability. If the characteristic equation is given by

$$a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n = 0$$

then the Routh-Hurwitz method constructs a table from the coefficients as follows:

$$\begin{array}{ccccc} s^n & a_0 & a_2 & a_4 & \ldots \\ s^{n-1} & a_1 & a_3 & a_5 & \ldots \\ s^{n-2} & X_1 & X_2 & X_3 & \ldots \\ s^{n-3} & Y_1 & Y_2 & Y_3 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ldots \end{array}$$

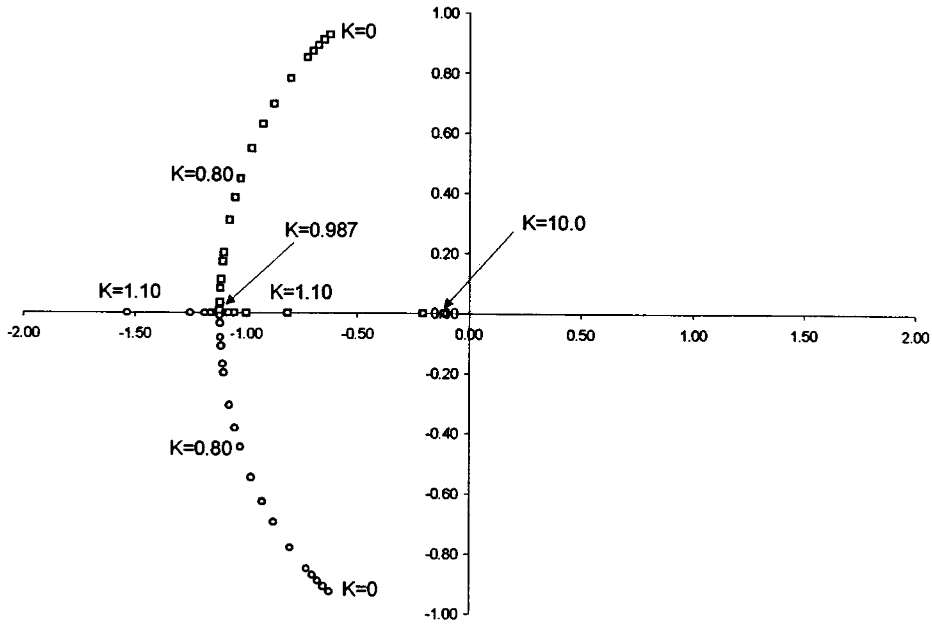


FIGURE 6.4.3 Root locus of $s^2 + (1.25 + K)s + 1.25 = 0$.

where

$$X_1 = \frac{a_1 a_2 - a_0 a_3}{a_1}; \qquad X_2 = \frac{a_1 a_4 - a_0 a_5}{a_1}; \qquad X_3 = \frac{a_1 a_6 - a_0 a_7}{a_1} \cdots$$

$$Y_1 = \frac{X_1 a_3 - a_1 X_2}{X_1}; \qquad Y_2 = \frac{X_1 a_5 - a_1 X_3}{X_1} \cdots$$

and so forth. The number of roots in the right-hand plane of the *s*-domain is equal to the number of sign changes in the first column, i.e., the column containing a_0, a_1, X_1, Y_1, etc. In other words, if all the elements in the first column have the same sign, then there are no roots in the right-hand plane and the process is stably controlled. Also, for special cases of the characteristic equation,

- If the first element of any row is zero but the remaining elements are not, then use some small value ε and interpret the final results as $\varepsilon \rightarrow 0$.
- If one of the rows before the final row is entirely zeros, then (1) there is at least one pair of real roots of equal magnitude but opposite signs, or (2) there is at least one pair of imaginary roots that lie on the imaginary axis, or (3) there are complex roots symmetric about the origin.

Field Commissioning — Installation, Calibration, Maintenance

Tuning of Feedback Loops

The *tuning* of a controller involves finding controller gains that will ensure at least a critically damped response of the process to a change in set point or process disturbance. A good starting point for PID constants is that derived during the design phase by the stability assessment approaches described above. However, real processes do not necessarily behave as their models would suggest and actual field tuning of controls is needed during the system-commissioning process.

Pole-Zero Cancellation. One method of obtaining the desired critically damped response of a process is to determine the closed-loop transfer function in the form

$$\frac{C}{R} = \frac{(s+A_1)(s+A_2)\ldots(s+A_m)}{(s+B_1)(s+B_2)\ldots(s+B_n)}$$

The coefficients *A* and *B* will depend on both the process characteristics and the controller gains. The objective of pole-zero cancellation is to find values for the controller gains that will set some numerator coefficients equal to those in the denominator, effectively canceling terms. As can be imagined, however, this can be a very difficult exercise, particularly when working with complex roots of the equations. This method can only be used with very simple system models.

Reaction Curve Techniques. Often it is advisable to test a feedback loop *in situ*. Several techniques have been developed that allow for the derivation of "good" PID constants for a given open-loop response. Consider the process response shown in Figure 6.4.4 where Δ_c is the change of process output, Δ_u is the change of controller, *L* is the time between change and intersection, and *T* is the time between lower intersection and upper intersection. We can define the following variables: $A = \Delta_u/\Delta_c$, $B = T/L$, and $R = L/T$. These values can be used with the equations given in Table 6.4.1 to estimate "decent" control constants. The users of these constants should be aware, however, that these constants are based on the typical response of second-order systems and may not provide good values for all processes.

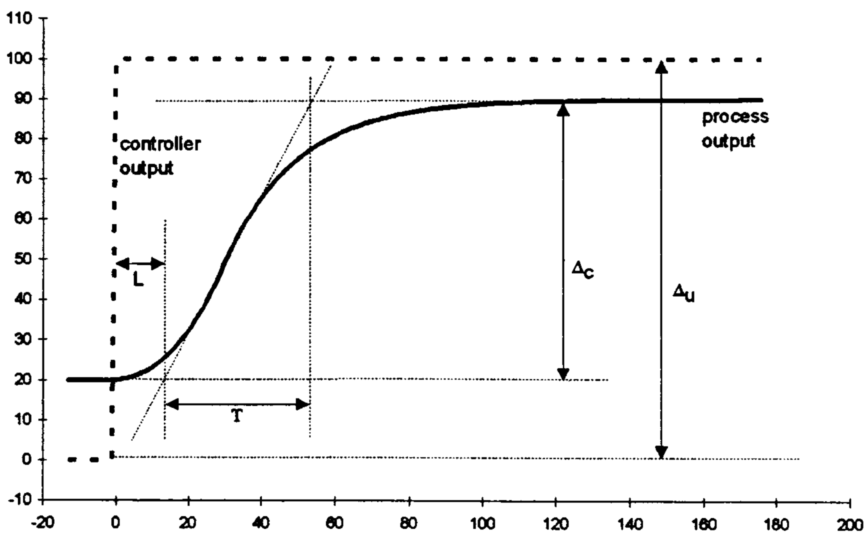


FIGURE 6.4.4 Reaction curve components.

Ultimate Frequency. The ultimate frequency test involves increasing the proportional gain of a process until it begins steady-state oscillations. K_p^* is defined as the proportional gain that results in steady oscillations of the controlled system and T^* is the period of the oscillations. The desired controller gains are given in Table 6.4.2. Note that the use of the ultimate period test is not always easy to do in practice and may be prohibited in certain cases by the a process operations manager.

TABLE 6.4.1 Equations for Finding PID Constants Using the Zeigler-Nichols and Cohen and Coon Reaction Curve Tests

	Zeigler-Nichols			Cohen and Coon		
Controller Components	K_p	$\frac{K_p}{K_i}$	$\frac{K_d}{K_p}$	K_p	$\frac{K_p}{K_i}$	$\frac{K_d}{K_p}$
P	AB	—	—	$AB\left(1+\frac{R}{3}\right)$	—	—
P + I	$0.9AB$	$3.3L$	—	$AB\left(1.1+\frac{R}{12}\right)$	$L\frac{30+3R}{9+20R}$	—
P + D	—	—	—	$AB\left(1.25+\frac{R}{6}\right)$	—	$L\frac{6-2R}{22+3R}$
P + I + D	$1.2AB$	$2L$	$0.5L$	$AB\left(1.33+\frac{R}{4}\right)$	$L\frac{32+6R}{13+8R}$	$L\frac{4}{11+2R}$

TABLE 6.4.2 Equations for Estimating PID constants Using the Ultimate Frequency Test

Controller Components	K_p	$\frac{K_p}{K_i}$	$\frac{K_d}{K_p}$
P	$0.5K_p^*$	—	—
P + I	$0.45K_p^*$	$0.8T^*$	—
P + I + D	$0.6K_p^*$	$0.5T^*$	$0.125T^*$

6.5 Advanced Control Topics

Peter S. Curtiss, Jan Kreider, Ronald M. Nelson, and Shou-Heng Huang

Neural Network-Based Predictive/Adaptive Controllers

Neural networks are powerful modeling tools used for predicting nonlinear behavior of processes and require a minimum of knowledge about the physical system involved. This approach can be used to predict the behavior of a process and can calculate the future value of the process variables. The effects of current modifications to the future value of the controlled process can be easily quantified and used to obtain the desired process response.

Overview of Neural Networks

The artificial neural network attempts to mimic a few aspects of the behavior of biological neural networks. Inputs to a biological nerve cell are carried along the dendrites of that cell. These inputs come from the positive impulse signals of other cells but may be converted to negative signals by the chemical interactions at the synapse between the cells. All of the inputs are then carried to the soma where they add or subtract from the overall potential difference between the interior of the soma and the surrounding fluid. Once the cell potential rises above a certain level, the cell "fires" and sends signals to other cells along its axon.

The artificial cell behaves in much the same way, except that the output signal is analog instead of digital. Signals from sending cells are passed along to a receiving cell through a series of connections. Each connection has an associated weighting factor that acts as a multiplier on the signal from the sending cell. All the inputs to a cell are summed (along with a cell bias, if included) and the resulting value is used to generate the output of the receiving cell. The output of the cell is referred to as the cell *activation* and the function that uses the net input to generate the cell activation is called the *activation function*. The activation function can theoretically be of any form, although linear and sigmoidal functions are frequently used. Figure 6.5.1 shows a comparison between a biological cell and an artificial cell.

When many different cells are combined together into a richly connected network (Figure 6.5.2), the result can behave mathematically like a nonlinear regression engine capable of mapping inputs to outputs for complex relationships. The trick is to find a series of weights W that allow the network to provide the desired outputs using specific inputs.

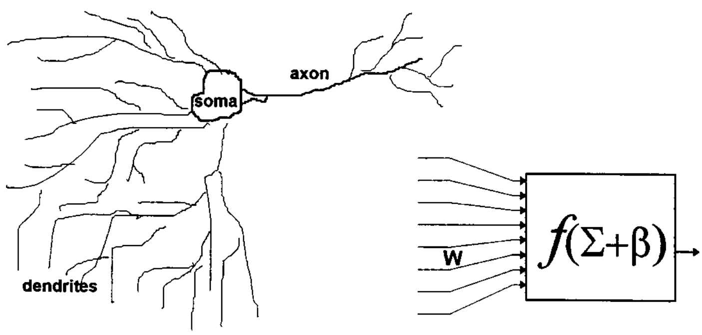


FIGURE 6.5.1 Biological cell vs. artificial cell.

Training Neural Networks

The net input to a cell is given by

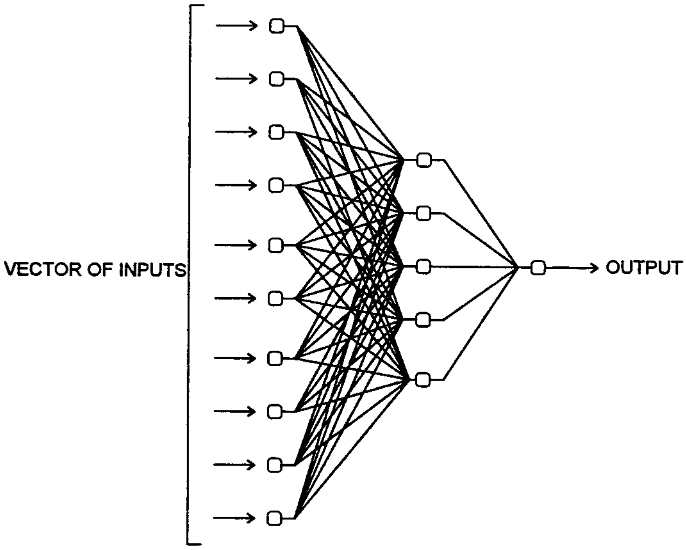


FIGURE 6.5.2 Artificial neural network consisting of several layers of mathematical models of biological neurons.

$$I_{\text{NET},i} = \sum_{j=f}^{1} A_j w_{j\to i}$$

where $I_{\text{NET,i}}$ is the net input of node *i* due to input from nodes *f* through *I* (as in *first* through *last*), A_j is the output of cell *j*, and $w_{j\to i}$ is the weighting factor associated with the connection between the sending node *j* and the receiving node *i*. The output (*activation*) of cell *i* is then going to be a function of the net input to the cell. Typically, the sigmoid function is used to relate activation to cell inputs

$$A_i = \frac{1}{1 + e^{-I_{\text{NET},i}}}$$

Practically any function can be used as the activation function of each node; the sigmoid function is usually chosen because the derivative, used during the training process, is easy to calculate. Traditional training of neural networks seeks to minimize the error function

$$E = \sum_{p=1}^{N_p} \sum_{i=1}^{N_o} \left(t_{p,i} - A_{p,i}\right)^2$$

where N_P is the number of input/output pairs of data, N_o is the number of outputs of the network, and $t_{p,i}$ is the desired (*target*) output value for a given set of inputs. This error function is minimized by adjusting the values of the weights proportionally to the negative of the derivative of the error with respect to each weight,

$$\Delta w_{j\to i} = -\varepsilon \frac{\partial E}{\partial w_{j\to i}}$$

where ε is the *learning rate* of the network. In a multilayered network the training is initiated by stimulating the network with a specific vector of inputs. The outputs of all the cells are then calculated

in order, starting with the input layer and ending with the output layer. The output is then compared with the known target output value, and any errors are compensated for by adjusting the weights from the output layer back toward the input layer. This method of training is, therefore. called *back-propagation*. With such a method, the previous equation for Δw can be rewritten as

$$\Delta w_{j \to i} = -\varepsilon \delta_i A_j$$

where δ_i represents the effect of a change in the net input to cell *j* on the output of cell *i*. For cells in the output layer,

$$\delta_i = \left(t_i - A_i\right) f'\left(I_{\text{NET},i}\right)$$

where f' is the derivative of the sigmoid activation function. For nodes in the hidden layers,

$$\delta_i = f'\left(I_{\text{NET},i}\right) \sum_{j=f}^{1} \left(\delta_i w_{j \to i}\right)$$

The product $d_i A_j$ is something called the *weight-error derivative* (WED). To prevent excessive oscillation of the weights during training, the change of weights can be restricted according to

$$\Delta w(k) = \varepsilon \cdot \text{WED} + \mu \cdot \Delta w(k-1)$$

where μ is the *momentum* of the network. Finally, to gradually reduce the rate at which the weights change, the learning rate ε can be made subject to exponential decay during the training process. This is sometimes called *simulated annealing*.

Bias nodes are like stand-alone cells that connect to each “normal” cell of the network. The output activation of each bias node is always unity and the “weight” connecting the bias node to the normal cell acts as a threshold function that suppresses or augments the output of the cell.

Using Networks for Controlling Feedback Loop Processes

Neural networks offer the potential for and have demonstrated improved control of processes through predictive techniques. The concept is fairly simple: train a network to predict the dynamic behavior of a process and then use these predictions to modify the controller output to place the process at a desired set point $R(t)$ at some time in the future. Initial results from computer simulations of such a controller are presented in Curtiss et al. (1993 a,b,c). Anderson (1989) described a computer simulation in which a network was trained to recognize the dynamic properties of an inverted pendulum (e.g., a broom balanced on an open palm). A control system was developed in which the angle and position of the pendulum were used to move the supporting base in order to maintain the pendulum upright. A neural network-based predictive controller is outlined in the classic discussion by Nguyen and Widrow (1989) on the “truck backer-upper” problem in which a tractor-trailer is backed into position at a loading dock.

Properly tuned fixed-gain controllers will usually work over a relatively wide range of process operation provided that the external perturbations and influences are small or time invariant. With nonlinear processes, however, a conventional control algorithm can lead to unstable control if the gains were chosen for a range different from the current operating conditions.

Architecture of the Network

With the neural network approach it is possible to overcome these problems by using as many additional salient inputs (the *auxiliary* inputs) as necessary and by incorporating an inherently nonlinear model to accomplish the control objectives. The network is trained using examples of the time-dependent relationship between a value of the feedback and previous values of the feedback, the controller output and

the auxiliary inputs. An example of the network architecture required for this is shown in Figure 6.5.3. In practice there does not need to be a limit on the number of previous measurements of any of the inputs, although the final size of the network and the corresponding training time and memory requirements need to be taken into consideration.

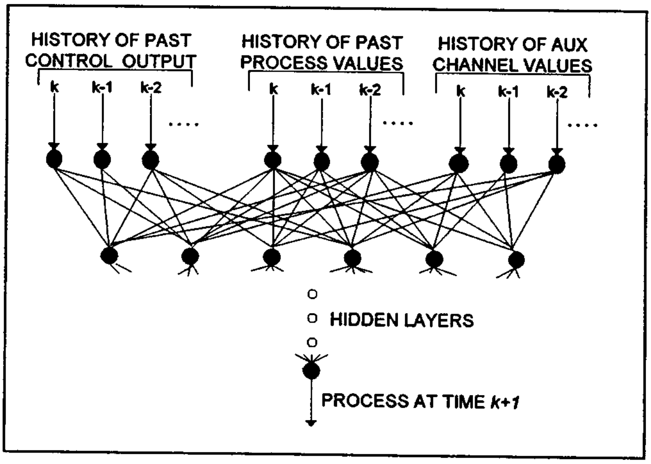


FIGURE 6.5.3 Network used for process prediction.

The network, once trained, can predict the future feedback value for any controller output. The trick is to find the controller output that causes the future process value to match the set point. This is accomplished by finding the derivative of the future error with respect to the current controller signal. Starting with the current process conditions, the feedback value is predicted at each time step into the future over a preset time window (). During each step of the prediction the values for the controller output and auxiliary inputs are held constant. This simulation is performed twice: the first time with a small increase in the controller output and the second time with a small decrease. This allows for the calculation of the change of the future process value (and hence the change of the future error) as a function of the change in the current controller output. The controller output is then modified by

$$\Delta U(t) = -G_{\text{net}} \cdot E_f(t) \frac{\partial E_f(t)}{\partial U(t)}$$

where E_f is the future error and G_{net} is the network controller gain. For a multiple-output controller, the additional outputs are simply added as more outputs of the network and the future predictions repeated several times to find the correct partial derivatives.

Many different variations on this theme are possible, for example, using the sum of the absolute values of all the errors over the prediction window (or the sum of the square of the errors, etc.) instead of simply the future error. Computer-simulated results of such tests are provided by Curtiss et al. (1993).

Estimating the Size of the Prediction Time Window. It is possible to use the network model to determine the size of the time window by estimating the amount of time required for the process to reach some future steady state after a simulated change in the controller output. An example of such an open-loop response it is shown in Figure 6.5.5. Here the network is simulating the response of a reverse-acting process after a decrease in actuator position at time step 0. About 70% (ln 2) of total rise time is achieved after 15 time steps. This kind of calculation can be performed during the control sequence and should indicate the proper time window size.

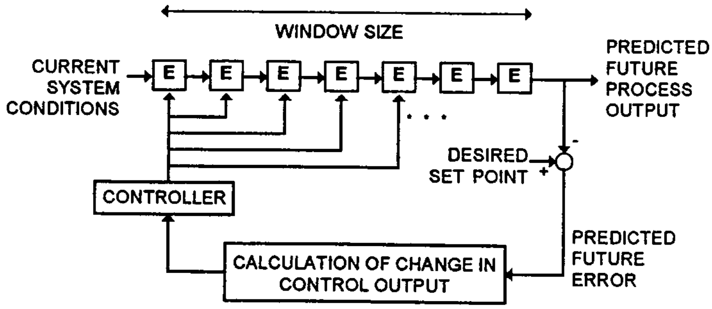


FIGURE 6.5.4 Schematic of procedure for determining future process value and error.

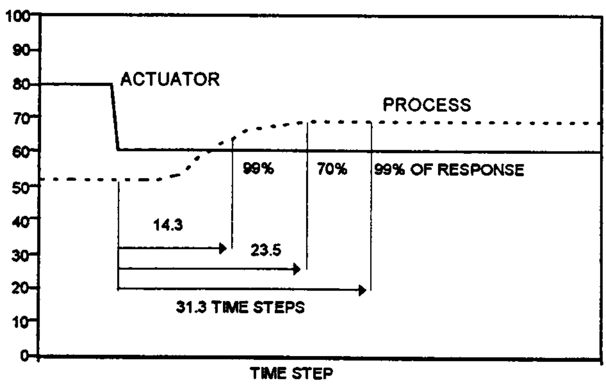


FIGURE 6.5.5 Example of computer-simulated process step change (used to determine size of time window).

Example of PID vs. Network Controller

Figure 6.5.6 shows an example of a process under PID control that demonstrates nonlinearity at different ranges of actuator position. Figure 6.5.7 shows the same process under the influence of a predictive neural network controller that had been trained on the process. Note that the network-based controller does not show the same problems of unstable control in certain actuator ranges. The size of the time window (15 time steps) was determined using the method discussed in the previous section. .

Using Networks as Supervisory Controllers

The previous section discussed the use of neural networks to minimize a predicted error of a feedback-loop process. It is possible to apply a similar methodology for supervisory plant control to optimize the process according to some cost function. A network is first trained to predict the cost function under a wide range of operating conditions. This network is then used to predict what will happen with different control strategies Figure 6.5.8. shows a schematic of this technique. The left side of the figure shows the training mode, where the network is attempting to associate the various plant inputs with the cost function output. There can be multiple inputs, including uncontrollable variables (e.g., ambient conditions, plant loads, etc.) and controlled variables (i.e., the various process set points.)

Once the network is sufficiently trained, it is used to find values for the set points under any set of uncontrolled variables. The technique for doing so is similar to the back-propagation training technique of the network. The inputs corresponding to the controlled variables are replaced with *virtual nodes* whose outputs are always unity. These nodes are connected to the predictor network through adjustable weights. The optimization occurs by finding values for these weights that allow the model to predict a desired output. These weights can be found through any number of search methods, including the gradient descent technique used in back-propagation training. In this case, the predictor network is “trained”

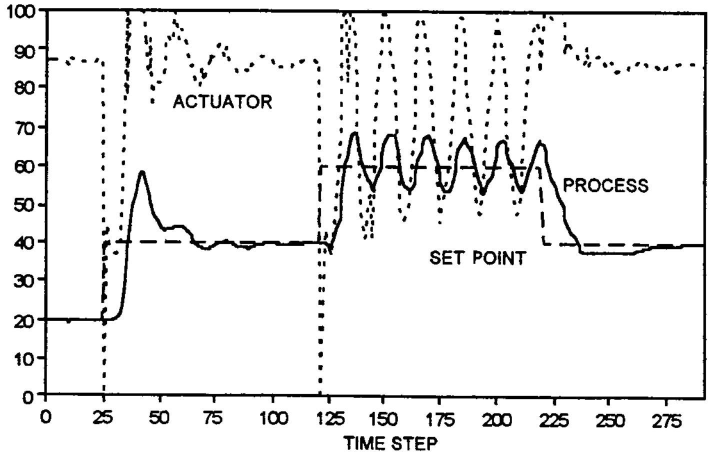


FIGURE 6.5.6 Example of computer simulation using a PID controller.

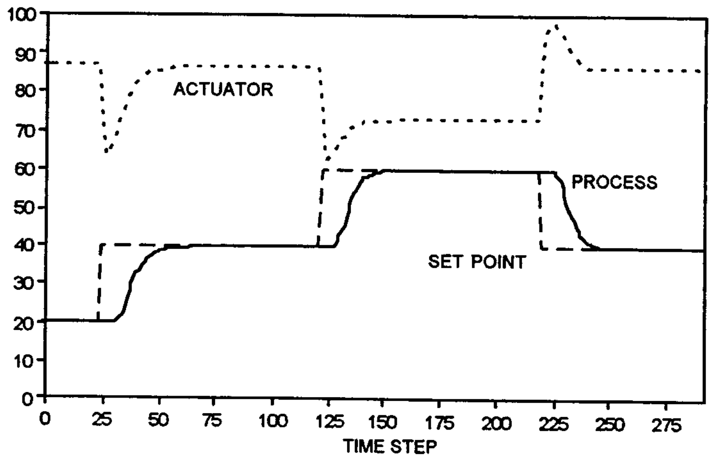


FIGURE 6.5.7 Example of computer simulation using a neural network controller.

normally, except that all weights in the network are static except those connected to the virtual nodes. Once weights have been found that produce the desired output, the set points can be found from direct interpretation of these weights. Constraints can be imposed on the weights either through physical limitations (e.g., freezing points) or from predictions from local-loop neural network controllers.

Fuzzy Logic Controllers

Fuzzy logic controllers (FLC) use conditional relationships to analyze one or more inputs. That is, the inputs are subject to a series of *if…then* queries to produce some intermediate values. An example would be something like a simple cruise control on an automobile:

- *If* vehicle speed = much lower than set point, *then* need to increase speed = large
- *If* vehicle speed = slightly lower than set point, *then* need to increase speed = small

These intermediate values are then used to determine the actual change of speed in the car:

- *If* need to increase speed = large, *then* increase of throttle position = 10%
- *If* need to increase speed = small, *then* increase of throttle position = 3%

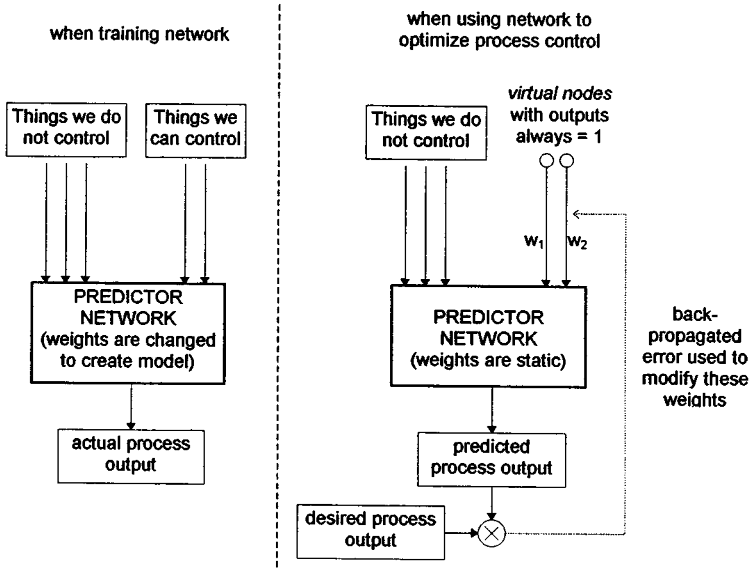


FIGURE 6.5.8 Using network model to optimize process control.

In fuzzy control, the satisfaction of a particular *if* statement may not lead to or be restricted by a true or false response. A range of weighting coefficients is assigned to a particular conditional, with the coefficients generally decreasing as the certainty of a specific condition decreases. In the example above, "need to increase speed = large" may be assigned a certainty of 1 when the speed of the vehicle is less than 50% of the desired speed of the car. This certainty will decrease as the speed of the car increases, so that "need to increase speed = large" may be 0 when the speed of the car is greater than 90% of the desired speed, but the certainty of "need to increase speed = small" will be large. It is possible that for a given speed of the car, two or more conditions may be satisfied with different magnitudes of certainty. These conditions can then be applied to the output rules along with their respective certainties to determine the actual increase (or decrease) in the controller output. When the speed of the car is below the desired speed, the initial rules may yield, for example,

- Need to increase speed = large with certainty = 0.3
- Need to increase speed = small with certainty = 0.7

the actual output would then be

- Increase of throttle position = $(0.3 \times 10\% + 0.7 \times 3\%)/(0.3 + 0.7) = 5.1\%$

The following section formalizes some of these ideas and includes a detailed example.

Section 19 Mathematics contains the formalism underlying fuzzy set theory and fuzzy logic. The reader is referred to that section and the one that follows for the technical basis for FLCs.

Fuzzy Logic Controllers for Mechanical Systems

Introduction

In the last decade, FLCs have been receiving more attention (Leigh and Wetton 1983; Daley and Gill, 1985; Yasunobu and Miyamoto, 1985; Xu, 1989), not only in test cases, but also in real industrial process control applications, including building mechanical systems (Sakai and Ohkusa, 1985; Ono et al., 1989; Togai and Maski, 1991; Huang and Nelson, 1991; Meijer, 1992). The basic idea of this approach is to incorporate the experience of human operators in the design of controllers. From a set of linguistic rules describing operators' control strategies, a control algorithm can be constructed (Ralston and Ward, 1985).

Computer simulations and experiments have shown that FLCs may have better performance than those obtained by conventional controllers. In particular, FLCs appear very useful when the processes are too complex for analysis using conventional control algorithms or when the available information is qualitative, inexact, or uncertain. Thus, fuzzy logic control may be viewed as a compromise between conventional precise mathematical control and humanlike decision making, as indicated by Gupta (Gupta and Tsukamoto, 1980).

However, fuzzy logic controllers sometimes fail to obtain satisfactory results with the initial rule set drawn from the operators' experience. This is because there still are some differences between the way a plant is operated by an experienced operator and by an FLC using the rules based directly on his or her experience. It is often difficult to express human experience exactly using linguistic rules in a simple form. Sometimes there is no experience that could be used to construct control rules for FLCs. In these cases it is necessary to design, develop, and modify control rules for FLCs to obtain optimal performance. There have been few discussions about rule development and adjustment strategies for FLCs (Sheridah, 1984; Scharf and Mandic, 1985; Wakileh and Gill, 1988; Ollero and Williams, 1989).

The Basic Aspects of an FLC

An FLC includes three parts: fuzzifier, fuzzy reasoning unit, and defuzzifier. The fuzzifier converts ordinary inputs into their fuzzy counterparts, the fuzzy reasoning unit creates fuzzy control signals based on these fuzzy variables, and the defuzzifier converts the fuzzy control signals into the real control outputs. The block diagram of a fuzzy control system is shown in Figure 6.5.9, where e, d, and u are tracking error, derivative error, and output control action; $\tilde{e}$, $\tilde{d}$, and $\tilde{u}$ are their fuzzy counterparts, respectively; y is the controlled parameter; and r is the set point for y. K_p is the scale factor for e, K_d is the scale factor for d, and K_o is the output gain.

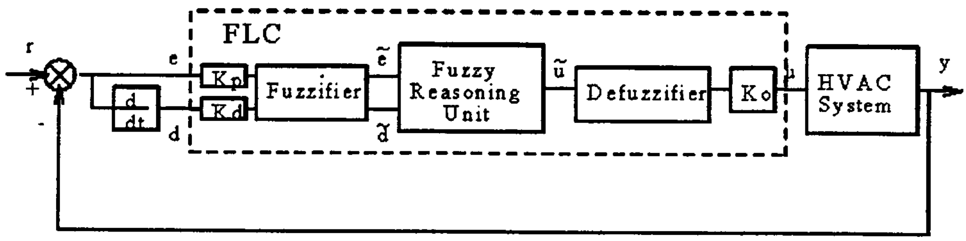


FIGURE 6.5.9 The block diagram of a fuzzy control system.

The control rules expressed in natural language are expressed in the following form:

IF (e is A) AND (d is B) THEN (u is C)

where A, B, and C are fuzzy subsets defined on the universes of discourse of e, d, and u, respectively. Every rule is interpreted into a fuzzy reasoning matrix:

$$R_k = \left[A_k(e) \otimes B_k(d)\right]^{\Theta} \otimes C_k(k) \qquad k = (1, N)$$

where N is the number of rules, the symbol ⊗ denotes aggregation operator, and the symbol Θ denotes an align-turning operator (see Section 19). The general fuzzy relation matrix R can be constructed as the union of the individual rules:

$$R = \bigcup_{k=1}^{N} R_k$$

This matrix represents the relationship between the fuzzy inputs and the fuzzy control output. The fuzzy control output can then be calculated from the known fuzzy input $\tilde{e}$ and $\tilde{d}$ by

$$\tilde{u} = \left[\tilde{e} \otimes \tilde{d}\right]^{\Theta} \circ R$$

where the symbol ° denotes the max-min composition operator (see Section 19).

The input universe of discourse for tracking error *e* or derivative error *d* is divided into several degrees connected with a number of fuzzy subsets by membership functions. In this study, *e* and *d* can each range from –6 to +6, and 13 degrees are used:

–6, –5, –4, –3, –2, –1, 0, 1, 2, 3, 4, 5, 6

Also, seven fuzzy subsets are defined as:

NL, NM, NS, ZZ, PS, PM, PL.

where the first letters N and P mean negative and positive, the second letters L, M, and S mean large, middle, and small, and ZZ means zero. These degrees and fuzzy subsets are shown in Table 6.5.1 which uses a 1.0–0.8–0.5–0.1 distribution. For example, if *e* = 3, then its membership in PL is 0.1, its membership in PM is 0.8, etc.

TABLE 6.5.1 The Membership Function of Input of an FLC

A(*e*), *B*(*d*)	–6	–5	–4	–3	–2	–1	0	1	2	3	4	5	6
PL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.5	0.8	1.0
PM	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.5	0.8	1.0	0.8	0.5
PS	0.0	0.0	0.0	0.0	0.0	0.1	0.5	0.8	1.0	0.8	0.5	0.1	0.0
ZZ	0.0	0.0	0.0	0.1	0.5	0.8	1.0	0.8	0.5	0.1	0.0	0.0	0.0
NS	0.0	0.1	0.5	0.8	1.0	0.8	0.5	0.1	0.0	0.0	0.0	0.0	0.0
NM	0.5	0.8	1.0	0.8	0.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NL	1.0	0.8	0.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

A similar analysis is given to the outputs for the control action indicated in Table 6.5.2 which uses a 1.0–0.7–0.2 distribution and where the abbreviations mean that the output control actions are Very Strong (Level 7), STrong (Level 6), SUbstrong (Level 5), MEdium (Level 4), Slightly Small (Level 3), SMall (Level 2), and TIny (Level 1).

TABLE 6.5.2 The Membership Function of Output of an FLC

C(*u*)	–6	–5	–4	–3	–2	–1	0	1	2	3	4	5	6
VS (Level 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.7	1.0
ST (Level 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.7	1.0	0.7	0.2
SU (Level 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.7	1.0	0.7	0.2	0.0	0.0
ME (Level 4)	0.0	0.0	0.0	0.0	0.2	0.7	1.0	0.7	0.2	0.0	0.0	0.0	0.0
SS (Level 3)	0.0	0.0	0.2	0.7	1.0	0.7	0.2	0.0	0.0	0.0	0.0	0.0	0.0
SM (Level 2)	0.2	0.7	1.0	0.7	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TI (Level 1)	1.0	0.7	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The fuzzifier converts ordinary inputs into their fuzzy counterparts. In this study, a fuzzy singleton is used as a fuzzification strategy, which interprets an input, *e* (or *d*), into a fuzzy value, $\tilde{e}$ (or $\tilde{d}$), with membership function (μ) equal to zero except at the element nearest to the real input, where μ = 1.0. For example, if *e* = 3.2, the nearest element is 3, then the fuzzy singleton will be

$$\tilde{d} = (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0)$$

This fuzzy singleton has membership function μ = 1.0 at the point of element e = 3. The defuzzifier converts the fuzzy control output created by the rule-based fuzzy reasoning unit into a real control action. In this study, weighted combination method is used as defuzzification strategy, which can be explained by the following example, if

$$\tilde{u} = (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.2,\ 0.4,\ 0.8,\ 0.7,\ 0.5,\ 0.1)$$

then

$$u = \left[0.2(1) + 0.4(2) + 0.8(3) + 0.7(4) + 0.5(5) + 0.1(6)\right] \big/ \left[0.2 + 0.4 + 0.8 + 0.7 + 0.5 + 0.1\right] = 3.4$$

Rule Refinement. An FLC is characterized by a set of linguistic statements which are usually in the form of *if-then* rules. The initial set of rules is usually constructed based on the operators' experience, or sometimes by analyzing the dynamic process of the controlled plant. Both approaches require modifying the initial set of rules to obtain an optimal rule set. This is called *rule refinement.*

Figure 6.5.10 shows an initial rule set analyzed on a "linguistic plane". The horizontal axis expresses the fuzzy subsets defined on the universe of discourse for the tracking error (e), and the vertical axis expresses the fuzzy subsets defined on the universe of discourse for the derivative error (d). Both have seven fuzzy "values": NL, NM, NS, ZZ, PS, PM, PL. On the cross points of these fuzzy values there are output control action levels, which are also fuzzy subsets having seven "values" from Level 1 (TIny) to Level 7 (Very Strong). For example, the cross point of e = NM and d = PM indicates u = Level 3. This corresponds to the rule:

IF (e is NM) AND (d is PM) THEN (u is Level 3)

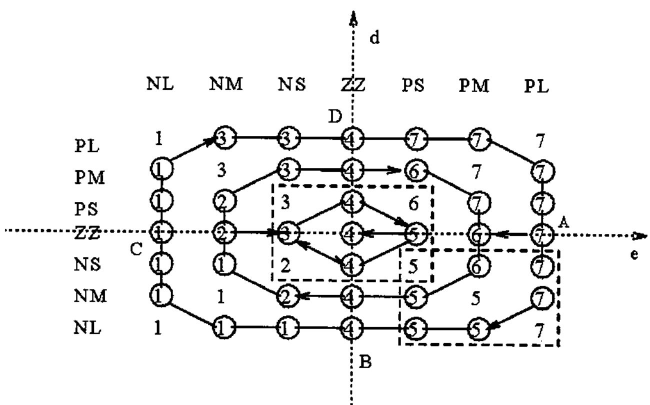


FIGURE 6.5.10 The initial rule set and performance trajectory on the linguistic plane.

For example, the initial rule set could be based on the following control strategies. First, it tries to keep a proportional relationship between the control action (u) and the tracking error (e). Note that if the derivative error (d) is ZZ, then the output control action (u) increases from Level 1 to Level 7 when the tracking error (e) changes from NL to PL. Second, the influence of derivative error (d) is considered such that, if it is positive, then increase the control action (u) a little bit, and if it is negative, then decrease the control action (u). For example, if the tracking error (e) keeps PM, the control action (u)

increases from Level 6 to Level 7 when the derivative error (d) is positive, and it decreases from Level 6 to Level 5 when the derivative error (d) is negative.

Consider a second-order plant with a transfer function:

$$H(s) = \frac{1.0}{s^2 + 0.1s + 1.0}$$

that is controlled using the initial rule set to respond to a step input for computer simulation. The performance trajectory of the FLC is shown by the arrows in Figure 6.5.10 and the dynamic process of the normalized controlled parameter (CP) is shown in Figure 6.5.11 where the horizontal axis indicates the number of sample period (SP). The dynamic process can be divided into two stages. At the first stage, there is a strong oscillation with a higher frequency, and, at the second stage, there is a moderate swing with a smaller frequency. Looking at the performance trajectory in the linguistic plane, we can see that the stronger oscillation occurs at the out-cycle (points further from the center). As time increases, the state moves to the in-cycle near the center of the plane and becomes moderate. This shows that FLCs have the desirable property of a structure-variable controller. The rules at the out-cycle belong to one kind of structure for the first stage, and the rules at the in-cycle belong to another structure for the second stage.

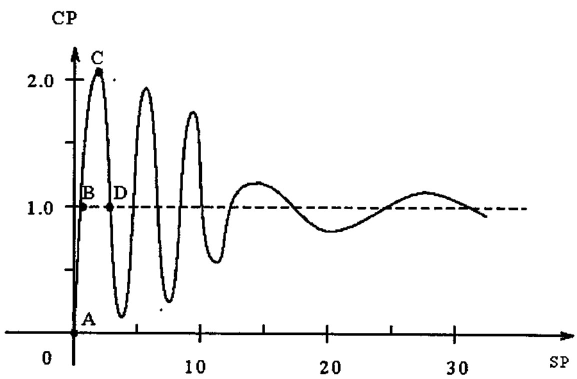


FIGURE 6.5.11 The dynamic process corresponding to Figure 6.5.10.

If the initial rule set does not satisfy a good design for a controller, then It can be modified by intuitive reasoning. A rule set is often symmetrically positioned about the central point, which is the desired stable operating point, where the tracking error (e) and the derivative error (d) both equal zero and the control action (u) is medium. When a positive step increase is imposed to the set point, the tracking error (e) has the biggest value and the derivative error (d) is zero at the beginning time (point A in the linguistic plane). With the regulating action, the tracking error (e) will decrease, the derivative error (d) will be negative, and the performance trajectory will enter into the right-bottom block in the linguistic plane. So, the rules in this area have the most important effect on the behavior of the first stage of the dynamic process. The most important area responsible for the behavior of the second stage is the central block.

To avoid strong oscillations, it is apparent that the control actions in the right-bottom block should be decreased. The modified rule set and its simulation of response to a step input are shown in Figure 6.5.12. The performance trajectory expressed in the linguistic plane is a spiral (Figure 6.5.12). We can see that the performance of the control system has been improved, but a small oscillation still exists and there is a little overshoot indicated by point C in Figures 6.5.12 to 6.5.13. Once again, the rule set is modified and the final rule set and its simulation of response to a step input are shown in Figure 6.5.14

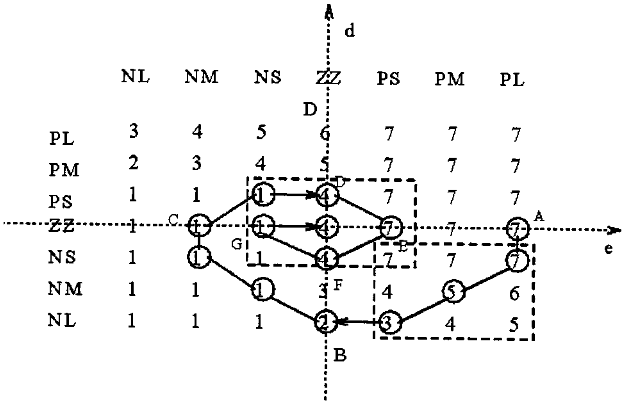


FIGURE 6.5.12 The second rule set on the linguistic plane.

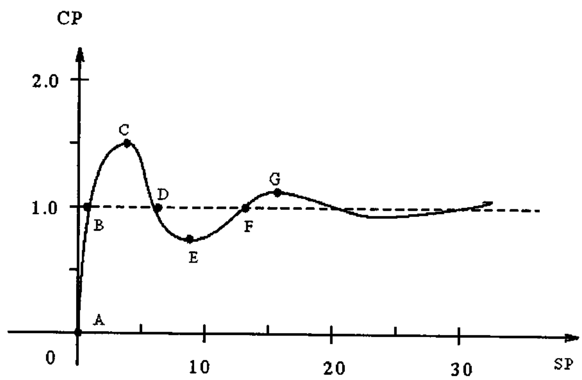


FIGURE 6.5.13 The dynamic process corresponding to Figure 6.5.12.

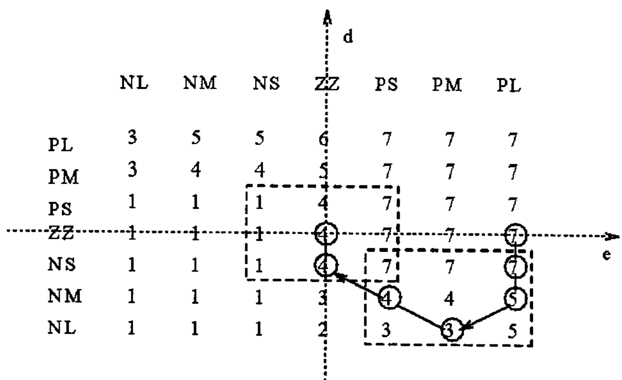


FIGURE 6.5.14 The third rule set on the linguistic plane.

and Figure 6.5.15. The final rule set gives good performance with a short rise time and a very small overshoot and it is considered satisfactory.

By analyzing the performance trajectory on the linguistic plane, a rule set is refined. It relies heavily on intuitive reasoning when comparing the dynamic process of the controlled parameter for the present rule set with the desired one.

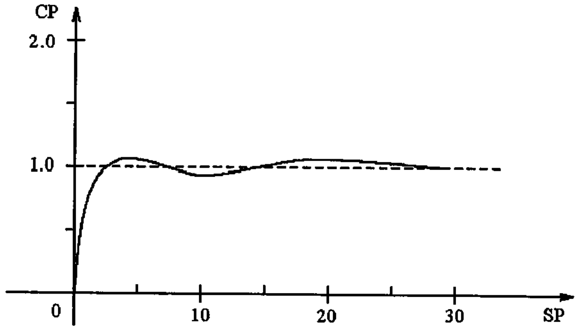


FIGURE 6.5.15 The dynamic process corresponding to Figure 6.5.14.

Completeness and Interaction of Rules and Selection of Membership Functions. The second significant influence on the behavior of an FLC is the membership functions. They should be chosen carefully in the adjustment process. As mentioned in Section 6.3, the fuzzy subsets, language variables NL, NM, NS, ZZ, PS, PM, and PL, are defined on the universe discourse of tracking error (e) or derivative error (d). Some possible membership functions are shown in Figures 6.5.16 to 6.5.18. The membership functions should be chosen to make these language variables have suitable coverage on the universe of discourse. For the case of Figure 6.5.16, the whole range is not covered by these language variables. There are some values of e or d, on which the membership functions of all language variables are zero. In this case, an empty output control action could be created. This means that the control actions are lost for those points which are not covered by any input fuzzy subset. This is referred as the non-completeness of control rules. FLCs should satisfy the condition of completeness for their membership functions. The membership function shown in Figure 6.5.17 cannot be used for an effective FLC. In other words, the union of all fuzzy subsets, X_i, i = [1,7], should be greater than zero for all $e \in E$, i.e.,

$$\forall e \in E \qquad \bigcup_{i=1}^{7} X_i(e) > 0$$

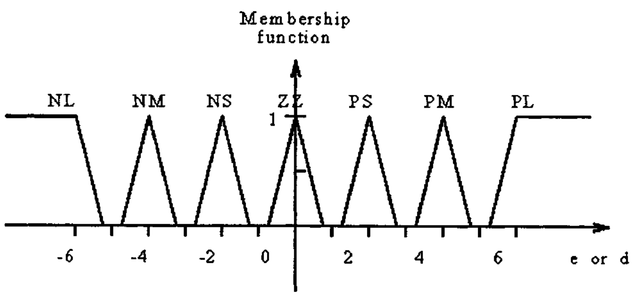


FIGURE 6.5.16 Non-complete membership function.

On the other hand, there can be interaction among the rules if the overlap of fuzzy subsets occurs on the range of the universe of discourse. In this case, the membership functions have the forms shown in Figures 6.5.17 and 6.5.18. The interaction tends to smooth out the set of control rules. Consider the single-input-single-output case for simplicity; the rule set is

IF (e is A_i) THEN (u is C_i) $\qquad i = [1,N]$

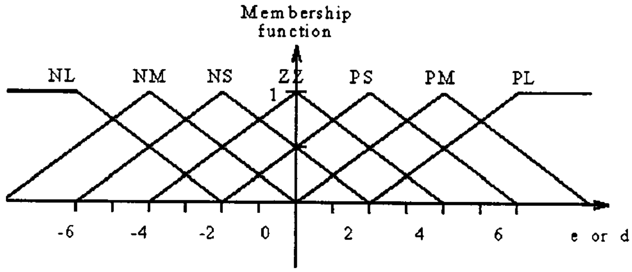


FIGURE 6.5.17 Heavy overlap membership function.

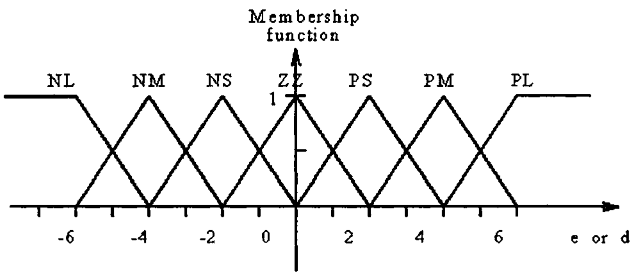


FIGURE 6.5.18 Moderate overlap membership function

where N is the number of rules in the set. These rules are incorporated into a fuzzy relation matrix as follows:

$$R = \bigcup_{i=1}^{N} R_i = \bigcup_{i=1}^{N} \left(A_i \otimes C_i \right)$$

If the fuzzy value of input e is known as $\tilde{e}$, the fuzzy output $\tilde{u}$ then can be calculated as follows:

$$\tilde{u} = \tilde{e} \circ R$$

If $\tilde{e}$ is A_i, $\tilde{u}$ is expected to be C_i. But now the interaction of rules due to overlap results in:

$$C_i \subseteq A_i \circ R$$

The equality is established only when no overlap occurs. This analysis is based on the fuzzy logic scheme including max-min composition operator. A more-detailed example of the numeric calculation is given in the appendices to this section.

If the overlap is heavy as shown in Figure 6.5.17, there will be large deformation and the control rules will lose their original shape. In the limit, as the membership functions become unity for all values, the output of the FLC will always be the same fuzzy quantity. This means that the fuzzy reasoning system conveys no valuable information and the FLC has lost its efficacy.

A moderate overlap, shown in Figure 6.5.18, is desirable to allow for reasoning with uncertainty and the need for completeness of the control rules. How does one determine the "size" of overlap? At present, we use intuitive judgment to choose membership functions when adjusting an FLC. There appears to be some latitude in choosing the amount of overlap, on which the performance of an FLC does not change significantly. The quantitative analysis will be given after further research.

When we modify the control rules in the linguistic plane, the overlapping membership functions let the rules near the performance trajectory have an effect on the output control actions. This is because interactions occur among the neighboring rules.

Scale Factors and Output Gain

The scale factors, K_p and K_d, and the output gain, K_o, shown in Figure 6.5.19, also have significant influence on the behavior of an FLC. Their influence is not as complicated as those of rules and membership functions. The adjustment for the scale factors and output gain is comparatively simple. The scale factor K_p relates the actual range of tracking error (e) to the universe of discourse (E) defined in the fuzzy logic system. In this work, E consists of 13 degrees as indicated in earlier sections. Then K_p is determined as the ratio of the range of E to the range of the real variable:

$$K_p = \frac{E_{\max} - E_{\min}}{e_{\max} - e_{\min}}$$

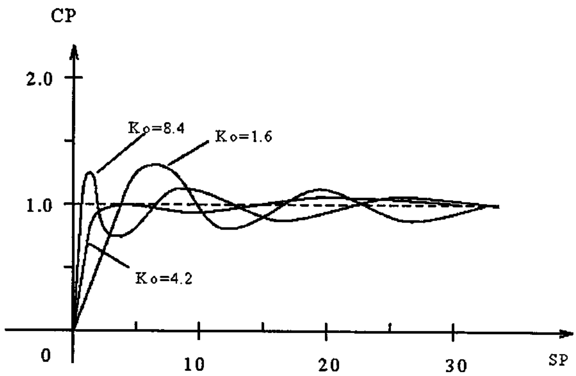


FIGURE 6.5.19 The influence of K_o on the behavior of FLCs.

For scale factor K_d, there is the similar analysis leading to

$$K_d = \frac{D_{\max} - D_{\min}}{d_{\max} - d_{\min}}$$

where D is the universe of discourse for derivative error (d) defined in the fuzzy logic system. Small K_p or K_d will narrow the control band, while large K_p or K_d will lead to loss of control for large inputs.

The output gain K_o is defined as follows:

$$K_o = \frac{u_{\max} - u_{\min}}{U_{\max} - U_{\min}}$$

It is the ratio of range of real output control action (u) to the range of its universe of discourse (U) defined in the fuzzy logic system. K_o acts as an amplification factor of the whole FLC. Figure 6.5.19 shows the influence of K_o on the step response simulation of an FLC with the final rule set used in Figure 6.5.14. Increasing K_o results in a shorter rise time. The performance trajectory in the linguistic plane will become steeper for the first stage and oscillation occurs. Decreasing K_o results in a longer rise time, and the performance trajectory in the linguistic plane will become moderate during the first stage. But, in our simulation, oscillation still occurred. This is because different K_o, larger or smaller, results in a new route of the performance trajectory which will activate the different rules which might

cause oscillation. So the influence of output gain, K_o, should be considered together with the change of the activated rules.

Conclusion

An FLC can perform much better than a conventional controller, such as a PID controller, if the FLC has been well constructed. The main disadvantage of using FLCs today seems to be the lack of a systematic procedure for the design of FLCs. The general method for designing an FLC is to use trial and observation. No useful mathematical tool has yet been developed for the design of an FLC because of its fuzziness, complexity, and nonparameterization.

There are three significant elements that have notable influence on the behavior of an FLC:

1. The control rules expressed in linguistic language,
2. The membership functions defined for fuzzy subsets, and
3. The scale factors attached to the input and the output gains.

The control rules play the main role in forming the dynamics of FLCs. The rule set can be analyzed and modified using the performance trajectory technique and evaluated using the dynamic process curve of the controlled parameter. The membership functions define the "shape" of fuzzy subsets. They should have appropriate width to avoid noncompleteness and suitable interaction among the fuzzy control rules. The scale factors (K_p and K_d) and output gain (K_o) serve as amplification factors.

At present, each application must be individually designed. The initial sets of rules are specifically set up for different applications. Work is now underway to develop a self-adaptive FLC which will choose the initial set of rules automatically according to the process dynamics and refine it on the basis of the global performance evaluation.

References

Anderson, C. 1989. Learning to control an inverted pendulum using neural networks, *IEEE Control Systems Magazine*, April, 31–36.

Askey, S. Y. 1995. Design and Evaluation of Decision Aids for Control of High Speed Trains: Experiments and a Model, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, June.

Billings, C.E. 1991. *Human-Centered Aircraft Automation: A Concept and Guidelines*, NASA Ames Research Center, Moffet Field, CA.

Curtiss, P.S., Kreider, J.F., and Brandemuehl, M.J. 1993a. Artificial neural networks proof of concept for local and global control of commercial building HVAC systems, in *Proceedings from the ASME International Solar Energy Conference*, Washington, D.C.

Curtiss, P.S., Kreider, J.F., and Brandemuehl, M.J. 1993b. Energy management in central HVAC plants using neural networks, in *Proceedings from the ASHRAE Annual Winter Meeting*, Chicago, IL.

Curtiss, P.S., Brandemuehl, M.J., and Kreider, J.F. 1993c. Adaptive control of HVAC processes using predictive neural networks, *ASHRAE Trans.*, 99(1).

Daley, S. and Gill, K.F. 1985. The fuzzy logic controller: an alternative design scheme?'' *Comput. Ind.*, 6, 3–14.

Gupta, M.M. and Tsukamoto, Y. 1980. Fuzzy logic controllers — a perspective.'' *Proc. Joint Automatic Control Conf.*, pp. FA10-C, August, San Francisco.

Huang, S.-H. and Nelson, R.M. 1991. A PID-law-combining fuzzy controller for HVAC applications, *ASHRAE Trans.*, 97(2), 768–774.

Huang, S.-H. and Nelson, R.M. 1994. Rule development and adjustment strategies of a fuzzy logic controller for an HVAC system: Part Two — experiment, *ASHRAE Trans*., 99(2), 851–856.

Leigh, R. and Wetton, M. 1983. Thinking clearly with fuzzy logic, *Process Eng.*, 64, 36–37.

MacArthur, J.W., Grald, E.W., and Konar, A.F. 1989. An effective approach for dynamically compensated adaptive control, *ASHRAE Trans.*, 95(2), 415-423.

Meijer, G. 1992. Fuzzy logic-controlled A/Cs heat pumps, *IEA Heat Pump Cent. Newslett.*, 10(1).

Nguyen, D.H. and Widrow, B. 1989. The truck backer-upper: an example of self learning in neural networks, *Proceedings of the International Joint Conference on Neural Networks*, 2, 357–363.

Ollero, A. and Williams, J. 1989. Direct digital control, auto-tuning, and supervision using fuzzy logic, *Fuzzy Sets Syst.*, 30, 135–153.

Ono, H., Ohnish, T., and Terada, Y. 1989. Combustion control of refuse incineration plant by fuzzy logic, *Fuzzy Sets Syst.*, 32, 193–206.

Radke, F. and Isermann, R. 1987. A parameter-adaptive PID-controller with stepwise parameter optimization, *Automatica*, 23, 449–457.

Ralston, P.A. and Ward, T.L. 1985. Fuzzy control of industrial process, in *Applications of Fuzzy Set Methodologies in Industrial Engineering*, Elsevier Science Publishers North-Holland; Amsterdam, 29–45.

Sakai, Y. and Ohkusa, K. 1985. A fuzzy controller in turning process automation, in *Industrial Application of Fuzzy Control*, Elsevier Science Publishers North-Holland; Amsterdam, 139–151.

Scharf, E.M. and Mandic, N.J. 1985. The application of a fuzzy controller to the control of a multi-degree-of-freedom robot arm, *Industrial Application of Fuzzy Control*, Elsevier Science Publishers North-Holland; Amsterdam, 1–18.

Sheridan, S.E. 1984. Automatic kiln control at Oregon Portland Cement Company's Durkee plant utilizing fuzzy logic, *IEEE Trans. Ind. Appl.*, 20, 562–568.

Sheridan, T.B. 1987. Supervisory control. In G. Salvendy, Ed., *Handbook of Human Factors/Ergonomics*, Wiley, New York.

Sheridan, T.B. 1992. *Telerobotics, Automation and Human Supervisory Control*, MIT Press, Cambridge, MA.

Shinners, S.M. 1978. *Modern Control System Theory and Application*, Addison-Wesley, Reading, MA.

Togai and Maski. 1991. An example of fuzzy logic control, *Comput. Des.*, 30, 93–103.

Wakileh, B.A. and Gill, K.F. 1988. Use of fuzzy logic in robotics, *Comput. Ind.*, 10, 35–46.

Xu, C.W. 1989. Fuzzy system identification, *IEEE Proc.*, 136, Pt. D, No. 4, pp 146–150.

Yasunobu, S. and Miyamoto, S. 1985. Automatic train operation system by predictive fuzzy control, in *Industrial Application of Fuzzy Control*, Elsevier Science Publishers North-Holland; Amsterdam, 1–18.

Further Information

The following are suggested reading for those interested in learning more about neural networks and their use in control systems:

Helferty, J.J., Collins, J.B., Wong, L.C., and Kam, M. 1989. A learning strategy for the control of a one-legged hopping robot, *Proceedings of the 1989 American Control Conference*, 896–901.

Kuperstein, M. and Rubinstein, J. 1989. Implementation of an adaptive neural network controller for sensory-motor coordination, *IEEE Control Syst. Mag.*, April, 25–30.

Lan, M. 1989. Adaptive control of unknown dynamical systems via neural network approach, *Proceedings of the 1989 American Control Conference*, 910–915.

Liu, H., Iderall, T., and Bekey, G. 1989. Neural network architecture for robot hand control, *IEEE Control Syst. Mag.*, April, 38–41.

Miller, R.C. and Seem, J.E. 1991. Comparison of artificial neural networks with traditional methods of predicting return time from night or weekend setback, *ASHRAE Trans.*, 97(2), 500–508.

Psaltis, D., Sideris, A., and Yamamura, A. 1988. A multilayered neural network controller, *IEEE Control Syst. Mag.*, April, 17–21.

Rumelhart, D.E. and McClelland, J.L. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, MA.

Wasserman, P.D. 1989. *Neural Computing: Theory and Practice*, Van Nostrand Reinhold, New York.

Appendices

Table of Transforms

The following table lists some of the more common transforms used in the analysis of building systems. More-extensive tables can be found in most mathematics and numerical analysis reference books.

List of Some *s*- and *z*-Transforms

Continous-Time Domain	Frequency Domain	Discrete-Time Domain
1 $t = 0$ 0 $t \neq 0$	—	1
1 $t = k$ 0 $t \neq k$	—	z^{-k}
1	$\frac{1}{s}$	$\frac{z}{z-1}$
t	$\frac{1}{s^2}$	$\frac{Tz}{(z-1)^2}$
e^{-at}	$\frac{1}{s+a}$	$\frac{z}{z-e^{-aT}}$
te^{-at}	$\frac{1}{(s+a)^2}$	$\frac{Tze^{-aT}}{\left(z-e^{-aT}\right)^2}$
$1 - e^{-at}$	$\frac{a}{s(s+a)}$	$\frac{z\left(1-e^{-aT}\right)}{(z-1)\left(z-e^{-aT}\right)}$
$e^{-at} - e^{-bt}$	$\frac{b-a}{(s+a)(s+b)}$	$\frac{z\left(e^{-aT}-e^{-bT}\right)}{\left(z-e^{-aT}\right)\left(z-e^{-bT}\right)}$

Special FLC Mathematical Operations

- $\otimes$ aggregation operator
- Θ align-turning operator
- $\circ$ max-min composition operator
- $\bigcup$ union operator
- $\exists$ exists
- $\in$ in
- $\forall$ for all
- $\subseteq$ is the subset of

The aggregation operator ($\otimes$) is used to define a two-dimensional fuzzy variable F from fuzzy subsets A and B (one-dimensional fuzzy variables) as follows:

$$F = A(e) \otimes B(d) = \begin{bmatrix} \mu_F(1,1) & \dots & \dots & \dots & \mu_F(1,N) \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \mu_F(i,j) & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \mu_F(M,1) & \dots & \dots & \dots & \mu_F(M,N) \end{bmatrix}$$

where

$$\mu_F(i,j) = \min\left(\mu_A(i), \mu_B(j)\right)$$

$$A(e) = \left(e_i, \mu_A(i)\right)$$

$$B(d) = \left(d_j, \mu_B(j)\right)$$

e_i is the ith element of the subset $A(e)$ and $\mu_A(i)$ is its membership function; d_j is the jth element of the subset $B(d)$ and $\mu_B(j)$ is its membership function.

The align-turning operator (Θ) is an operator acting on a two-dimensional fuzzy variable to create a one-dimensional fuzzy variable, which has a set of membership functions aligned according to a certain order, as follows:

$$\begin{aligned} S &= \left[A \otimes B\right]^{\Theta} \\ &= \left(\mu_S(1,1), \mu_S(1,2), \ldots, \mu_S(i,j), \ldots, \mu_S(M,N)\right) \end{aligned}$$

where j varies from 1 to N first and i varies from 1 to M.

An Example of Numeric Calculation for Influence of Membership Function

Suppose that there are two rules:

R1: IF (e is PM) THEN (u is VS) and

R2: IF (e is PS) THEN (u is ST).

For the first rule,

$$\tilde{e}_1 = (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.1,\ 0.5,\ 0.8,\ 1.0,\ 0.8,\ 0.5)$$

$$\tilde{u}_1 = (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.2,\ 0.7,\ 1.0)$$

Then the rule can be interpreted into a fuzzy reasoning matrix as follows:

$$R_1 = \tilde{e}_1 \otimes \tilde{u}_1$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.5 \end{bmatrix}$$

$$\equiv \mu_{R_1}(i,j)$$

where the membership in R_1 for the element (i,j) of the matrix, $\mu_{R_1}(i,j)$ is

$$\mu_{R_1}(i,j) = \min\left(\mu_{\tilde{e}_1}(i), \mu_{\tilde{u}_1}(j)\right)$$

For the second rule,

$$\tilde{e}_2 = (0,\ 0,\ 0,\ 0,\ 0,\ 0.1,\ 0.5,\ 0.8,\ 1.0,\ 0.8,\ 0.5,\ 0.1,\ 0)$$

$$\tilde{u}_2 = (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.2,\ 0.7,\ 1.0,\ 0.7,\ 0.2)$$

Then the second rule can be interpreted into a fuzzy reasoning matrix as follows:

$$
\begin{aligned}
R_2 &= \tilde{e}_2 \otimes \tilde{u}_2 \\
&= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.5 & 0.5 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 0.8 & 0.7 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 1.0 & 0.7 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 0.8 & 0.7 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.5 & 0.5 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \\
&\equiv \mu_{R_2}(i,j)
\end{aligned}
$$

where the membership in R_2 for the element (i,j) of the matrix, $\mu_{R_2}(i,j)$ is

$$
\mu_{R_2}(i,j) = \min\left(\mu_{\tilde{e}_2}(i), \mu_{\tilde{u}_2}(j)\right)
$$

The general fuzzy relation matrix R is then constructed as the union of these two rules:

$$
\begin{aligned}
R_2 &= R_1 \cup R_2 \\
&= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.5 & 0.5 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 0.8 & 0.7 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 1.0 & 0.7 & 0.5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.7 & 0.8 & 0.7 & 0.8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.5 & 0.7 & 1.0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.1 & 0.2 & 0.7 & 0.8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.5
\end{bmatrix} \\
&\equiv \mu_R(i,j)
\end{aligned}
$$

where

$$\mu_R(i,j) = \max\left(\mu_{R_1}(i), \mu_{R_2}(j)\right)$$

Assume that there is an input (e) and its fuzzy value $\tilde{e}$ = PM, then the output is expected to be VS according to the first rule. But now the output is calculated through the fuzzy matrix R as follows:

$$\begin{aligned}\tilde{u} &= \tilde{e} \circ R \\ &= \text{PM} \circ R \\ &= (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.1,\ 0.5,\ 0.8,\ 1.0,\ 0.8,\ 0.5) \circ R \\ &= (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.2,\ 0.7,\ 0.8,\ 0.7,\ 1.0)\end{aligned}$$

where

$$\mu_{\tilde{u}}(j) = \max_i\left(\min\left(\mu_{\tilde{e}}(i), \mu_R(i,j)\right)\right)$$

While

$$\begin{aligned}\tilde{u}_1 &= \text{VS} \\ &= (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.2,\ 0.7,\ 1.0)\end{aligned}$$

So,

$$\tilde{u}_1 \subseteq \tilde{u}$$